UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

COMPLEJIDAD COMPUTACIONAL EN VIDEOJUEGOS

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA EN COMPUTACIÓN

PRESENTA:

JOSÉ RICARDO ROSAS BOCANEGRA

DIRECTOR DE TESIS:
DR. JOSÉ DAVID FLORES PEÑALOZA

FACULTAD DE CIENCIAS

Ciudad Universitaria. CDMX. Enero 2021

Agradecimientos

Agradezco infinitamente a mis padres y mis hermanos, que siempre han estado apoyándome y gracias a los cuales he llegado hasta donde estoy ahora.

Agradezco el apoyo otorgado por CONACYT, que fue indispensable durante toda la maestría.

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM IN120520. Agradezco a la DGAPA-UNAM la beca recibida.

Índice general

Índice de figuras

Resumen

El objetivo de este trabajo es presentar una variedad de problemas de decisión encontrados en videojuegos, analizando su complejidad y demostrar que son *NP-Duros*, *NP-Completos*, *PSPACE-Duros* o *PSPACE-Completos*.

La motivación principal de este trabajo es extender los resultados previos de esta línea de estudios. Estos resultados fueron publicados en los artículos [27], [7], [1], [4], [8], [9] y [5].

Este trabajo está organizado de la siguiente forma:

En el primer capítulo se presentan los fundamentos teóricos esenciales para el entendimiento del resto de este trabajo. Se explica de manera detallada las definiciones necesarias sobre Máquinas de Turing, complejidad computacional y clases de complejidad. Se dan las definiciones generales de las clases de complejidad más comunes así como la relación que existe entre ellas. También se definen términos como dureza, completitud y reducciones polinomiales.

En el segundo capítulo se presentan problemas de decisión en videojuegos estudiados en trabajos previos demostrados como *NP-Duros*, *NP-Completos*, *PSPACE-Duros* o *PSPACE-Completos*. Se presentan reglas generales de cada uno, mismas que se utilizan en las demostraciones. Finalmente se presentan las demostraciones de dureza o completitud.

En el tercer capítulo se presentan problemas de decisión en videojuegos que no han sido estudiados en trabajos anteriores. Al igual que en el capítulo anterior, se presentan primero las reglas generales de los videojuegos estudiados. Posteriormente se presentan las demostraciones de dureza o completitud.

Finalmente, en el cuarto capítulo se expondrán las conclusiones finales de este trabajo. Además, se plantea el posible trabajo a futuro a realizar relacionado con este trabajo.

1 Introducción

Este trabajo se centrará en el análisis de problemas de decisión. En general, una gran cantidad de problemas a resolver pueden verse como un problema de decisión. Estos problemas pueden expresarse de manera informal como "dado en elemento y una propiedad fija, ¿el elemento cumple con la propiedad?".

En esta sección se dará una descripción general de algunos términos que serán necesarios para el propósito general de este trabajo.

1.1. Lenguajes formales

Para el análisis de los problemas de decisión, es necesario definir el término lenguaje formal.

Tomando la definición de[13], un lenguaje formal es un conjunto de cadenas seleccionadas sobre Σ^*, donde Σ es un alfabeto. A su vez, se puede definir un alfabeto como un conjunto finito de símbolos, y Σ^* es el conjunto de todas las cadenas posibles de un alfabeto Σ. Por otro lado, una cadena es una secuencia finita de símbolos seleccionados de un alfabeto Σ.

Si Σ es un alfabeto, L un lenguaje y $L \subseteq \Sigma^*$, se dice que L es un lenguaje sobre Σ. Dado que L no necesita contener todos los símbolos de Σ, si L es un lenguaje sobre Σ, con Σ y Σ' alfabetos, y $\Sigma \subseteq \Sigma'$, entonces L también es un lenguaje sobre Σ'.

Una vez definido un lenguaje, se puede considerar a un problema de decisión como el de decidir si una cadena x pertenece al lenguaje L, de manera más formal, si $f(x)$ es la función booleana (que corresponde al problema de decisión), el lenguaje asociado L_f se define como $L_f = \{x : f(x) = 1\}$.

1.2. Máquinas de Turing y algunas variantes

Las Máquinas de Turing son un modelo computacional poderoso y sencillo, que permite analizar todo lo que es computable. Es importante analizarlas para entender la noción de tiempo de ejecución y a su vez, de complejidad computacional.

De manera informal, una Máquina de Turing (abreviado MT) con k cintas se compone de dos cosas[2]:

- Área de trabajo: Esta área de trabajo consiste en k cintas. Una cinta es una línea de celdas, de longitud no acotada hacia una dirección (en general y sin pérdida de generalidad, de longitud no acotada hacia la derecha). Cada celda puede contener un símbolo del alfabeto Γ, el cual es llamado alfabeto de la MT.

  Cada cinta tiene una cabeza que puede escribir o leer símbolos de la celda y moverse a la izquierda o a la derecha. Los cálculos de la MT se discretizan de tal manera que en cada paso la cabeza de cada celda puede leer y escribir un símbolo, y moverse una casilla a la izquierda o a la derecha.

  La primera cinta de la MT es llamada cinta de entrada, y es de sólo lectura, no se puede escribir en ella. Las otras $k-1$ cintas son llamadas las cintas de trabajo, son de lectura y escritura. La última de estas $k-1$ cintas es llamada la cinta de salida, en esta se encuentra el resultado final cuando la MT se detiene, si es que lo hace.

- Estados finitos de las operaciones y reglas: La MT tiene un conjunto finito Q de estados. Se tiene un registro en el cual se guarda un elemento de Q, este es el estado actual de la MT. A partir de cada estado, se define las acciones a realizar en cada paso. Estas acciones, a las que intuitivamente se les llamará reglas, se definen con la función de transición que se explicará más adelante.

  Las reglas se componen de cuatro cosas: Los símbolos leídos en la cabeza de cada una de las k cintas; el símbolo que se escribirá en donde se encuentra la cabeza de cada una de las $k-1$ cintas de trabajo; el estado siguiente que reemplazará al actual en el registro (puede elegir el mismo estado); y la dirección en la que se moverá la

cabeza de cada una de las k cintas. En la figura 1.1 se puede observar un ejemplo de MT.

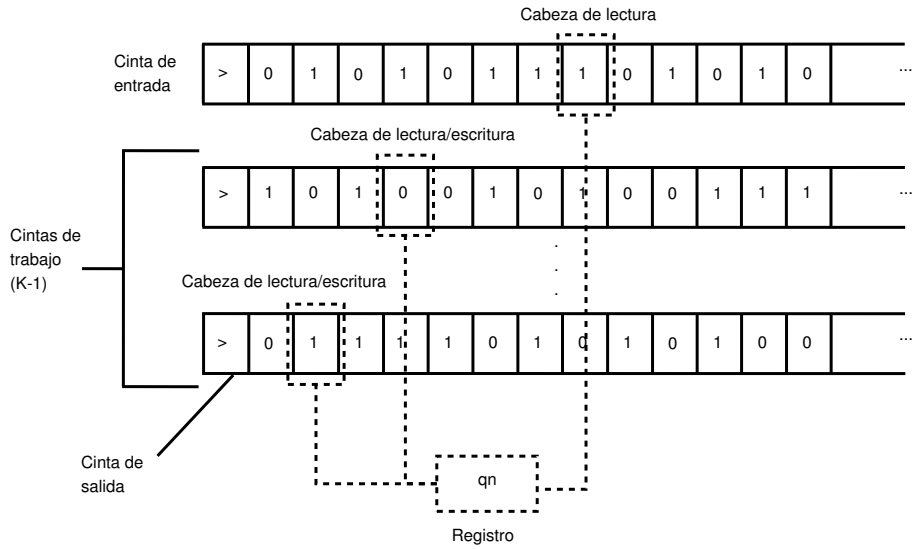


Figura 1.1: Máquina de Turing de k cintas, una cinta de entrada y $k-1$ de trabajo, siendo la última la cinta de salida.

Siguiendo la definición de[2], formalmente, una MT M se define como una tupla (Γ, Q, δ). Los elementos de esta tupla son:

- Un conjunto Γ de símbolos que las cintas de M pueden contener. Este conjunto es llamado alfabeto. En general se asume que el alfabeto contiene un símbolo de inicio $\triangleright$, un símbolo vacío $\square$ los símbolos 0 y 1.

- Un conjunto Q de estados posibles de M. En general se supone que Q contiene al menos un estado de inicio q_{inicio} y uno de paro q_{paro}. En algunas definicios se utilizan los estados especiales $q_{aceptar}$ y $q_{rechazo}$, los cuales sirven para definir cuando una cadena es un elemento o no de un lenguaje, pero no son necesarios en esta definición.

- Una función $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{I, A, D\}^k$ que describe las reglas a seguir en cada paso, $\{I, A, D\}$ siendo las direcciones a las que puede moverse cada cabeza, I izquierda, A quedarse en posición actual y D derecha. Esta función se llama función de transición de M.

Teniendo en cuenta la definición formal, hace falta entender la forma en que este modelo permite realizar cómputos, para ello se definirán los conceptos de configuración inicial y la forma en que trabaja su función de transición.

La MT tiene una configuración con la que se empiezan sus cómputos, esta es llamada configuración inicial. Esta configuración inicial es la siguiente:

- Cada cinta de trabajo contiene un símbolo de inicio en la celda del final izquierdo de la cinta, seguido de símbolos vacíos en todas las otras celdas.

- La cinta de entrada contiene un símbolo de inicio en la celda del final izquierdo de la cinta, seguido de una cantidad finita de símbolos no vacíos (que son la entrada de la MT), y finalmente símbolos vacíos en todas las otras celdas.

- La cabeza de todas las cintas se encuentran en el final izquierdo de su cinta correspondiente. El estado actual es q_{inicio}.

Como se dijo en su definición, la función de transición de M toma un estado y k símbolos del alfabeto, esto indica que se tiene definida la acción a realizar dado el estado en que se encuentre la MT y el símbolo que la cabeza de cada cinta lea.

La función devuelve un estado que es el estado siguiente que reemplazará al actual en el registro; $k-1$ símbolos que son los que se escribirá la cabeza de cada una de las $k-1$ cintas de trabajo en su posición; y k direcciones que indican hacia donde se mueve la cabeza de cada cinta (si la cabeza se encuentra en el borde a la izquierda y se trata de mover en esa dirección, la cabeza se queda en su posición actual).

Un paso de cómputo de la MT se realiza aplicando la función de transición una vez. Se dice que la MT se ha detenido cuando llega al estado q_{paro}, cuando se llega a este estado, la función de transición no permite avanzar más en la ejecución.

Una vez definida una MT con k cintas, es sencillo definir una MT con una sola cinta. Una MT de una cinta se define utilizando la misma cinta como cinta de entrada y cinta de trabajo, y definiendo una sección de la cinta

como la salida. Adicionalmente, la función de transición queda definida como $\delta : Q \times \Gamma \to Q \times \Gamma \times \{I, A, D\}$.

La definición de MT permite medir el tiempo de ejecución de su cómputo dado que cada paso está bien definido. Ya que todo algoritmo necesita al menos leer la entrada, se definirá el tiempo de ejecución como una función del tamaño de la entrada. La definición formal, utilizada en[2], es la siguiente:

Sean $f : \{0,1\}^* \to \{0,1\}^*$ y $T : \mathbb{N} \to \mathbb{N}$ dos funciones, y M una MT. Se dice que M computa f en tiempo $T(n)$ si para cada $x \in \{0,1\}^*$, si M está inicializada con configuración inicial con la entrada x, y después de a lo más $T(|x|)$ pasos se detiene con $f(x)$ como salida.

Se dice que M computa f si computa f en tiempo $T(n)$ para alguna función $T : \mathbb{N} \to \mathbb{N}$.

Por otro lado, se dice que una función $T : \mathbb{N} \to \mathbb{N}$ es tiempo construible si $T(n) \geq n$ y existe una MT M que compute la función $x \mapsto \llcorner T(|x|) \lrcorner$ en tiempo $T(n)$ (la expresión $\llcorner T(|x|) \lrcorner$ denota la representación binaria de $T(|x|)$). Algunos ejemplos de funciones tiempo construibles son n, $n \log(n)$, n^2 y 2^n.

El uso de una MT con una o varias cintas, o con un alfabeto grande o pequeño, no influye significativamente en los resultados de este trabajo.

En[2] se menciona y se muestra un bosquejo de la demostración de que para toda M MT, con alfabeto Γ que computa una función f en tiempo $T(n)$, es posible construir una MT M' con alfabeto $\{\triangleright, \square, 0, 1\}$ que compute f en $4 \cdot \log|\Gamma| \cdot T(n)$. También se menciona y se muestra un bosquejo de la demostración en[2] de que para toda M MT, con k cintas que computa una función f en tiempo $T(n)$, se puede construir una MT M' con una cinta que compute f en $5 \cdot k \cdot T(n)^2$.

Por otro lado, otra variante MT importante es la llamada MT No Determinista. Como se puede ver en[2], una MT No Determinista (abreviadas como MTND) tiene casi las misma características que las de una MT estándar, con la diferencia de que la MTND cuentan con dos funciones de transición, δ_1 y δ_2. Además, tiene un estado especial $q_{aceptar}$.

Al computar una función, la MTND ejecuta arbitrariamente una de las

funciones de transición. Se dice que una MTND M acepta una entrada x (en otras palabras, que $M(x) = 1$) si al ejecutarse con ella, existe al menos una secuencia en la cual M llega al estado $q_{aceptar}$, y al contrario, $M(x) = 0$ si para toda secuencia en la ejecución, nunca se llega al estado $q_{aceptar}$.

Se dice que M se ejecuta en tiempo $T(n)$ si para toda secuencia de decisiones no deterministas, M llega al estado $q_{aceptar}$ o q_{paro} con la entrada x en a lo más $T(|x|)$ pasos.

A diferencia de las otras variantes mencionadas, no se ha podido comprobar si en general, siempre se puede construir una MT estándar a partir de una MTND sin que tenga un impacto fuerte en su tiempo de ejecución.

1.3. Clases de complejidad

1.3.1. Definición de clases de complejidad

Como se indica en[2], una clase de complejidad es un conjunto de funciones que pueden ser computadas con un recurso dado. En este trabajo, al igual que en muchos otros, se estudiarán funciones booleanas con salida de un solo bit, las cuales también son llamadas *problemas de decisión*. Como se mencionó anteriormente, dado que los problemas de decision se puede identificar con el lenguaje $L_f = \{x : f(x) = 1\}$, se puede nombrar a estas como funciones o lenguajes indistintamente.

Esta definición permite clasificar problemas dependiendo del tiempo de ejecución necesario para resolverlo. Como se puede observar, la definición de clases de complejidad no impide que un problema sea parte de más de una clase de complejidad a la vez. Para un mejor entendimiento de la definición, se definirá la clase *DTIME* para las MT, siguiendo la definición de[2].

Sea $T : \mathbb{N} \to \mathbb{N}$ una función, la clase $DTIME(T(n))$ se define como el conjunto de todas las funciones booleanas con salida de un bit que son computables por alguna MT en $c \cdot T(n)$, para alguna constante $c > 0$.

Análogamente, se define en[2] la clase *NTIME* para MTND de la siguiente manera:

Sea $T : \mathbb{N} \to \mathbb{N}$ una función y $L \subseteq \{0,1\}^*$ un lenguaje, $L \in NTIME(T(n))$ si existen una constante $c > 0$ y una MTND de tiempo $c \cdot T(n)$ tal que para toda $x \in \{0,1\}^*, x \in L$ si y sólo si $M(x) = 1$.

Con la definición de *DTIME* se puede definir la clase P como sigue:

$$P = \bigcup_{c \geq 1} DTIME(n^c).$$

Esta definición indica que la clase P contiene a las funciones que pueden ser computadas en un tiempo que puede ser expresado como un polinomio. Si se considera que un problema de decisión se *resuelve* al computar la función, puede interpretarse como que la clase P contiene a todos los problemas de decisión que pueden ser resueltos en tiempo polinomial.

Por otro lado, otra clase de complejidad importante es la clase *NP*. Intuitivamente, la clase *NP* contiene a las funciones cuyas instancias que se mapean a 1 son rápidamente verificables (en específico, pueden verificarse en tiempo polinomial).

Existen varias definiciones formales de la clase *NP*, en este trabajo se utilizará la siguiente mencionada en [2]:

Un lenguaje $L \subseteq \{0,1\}^*$ está en *NP* si existe un polinomio $p : \mathbb{N} \to \mathbb{N}$, y una MT de tiempo polinomial M, tal que para toda $x \in \{0,1\}^*$, $x \in L$ si y sólo si existe $u \in \{0,1\}^{p(|x|)}$ tal que $M(x,u) = 1$.

Si $x \in L$ y $u \in \{0,1\}^{p(|x|)}$ satisfacen $M(x,u) = 1$, se dice que u es un certificado de x.

Esta definición, intuitivamente, dice que un lenguaje L está en *NP* si dada una $x \in L$, existe un certificado de tamaño polinomial de que x es efectivamente elemento de L, y este certificado puede verificarse en tiempo polinomial.

Un ejemplo conocido de un problema en *NP* es el problema *SAT*. Este problema consiste en decidir si dada una expresión booleana, existe una asignación de valores que la satisface. En este caso, el lenguaje asociado al problema es el lenguaje conformado por las expresiones booleanas que son satisfacibles.

Para este ejemplo, el certificado es simplemente la asignación de las variables que satisface a la expresión, pues esta es de tamaño polinomial y basta con asignar los valores a las variables para confirmar que la expresión se satisface, lo cual tiene un tiempo de ejecución polinomial con respecto al tamaño de entrada.

La clase *NP* también puede definirse utilizando la clase *NTIME*, esta definición, tomada de[2], servirá para contrastar algunas definiciones posteriores de otras clases, pero en general en este trabajo se utilizará la primera. La definición de *NP* utilizando la clase *NTIME* es la siguiente:

$$NP = \bigcup_{c \geq 1} NTIME(n^c).$$

En secciones posteriores se profundizará sobre la relación que existe entre las clases *P* y *NP*, el cual es un tema de gran importancia y discusión en el área de la teoría de la complejidad. Antes de profundizar en este tema, se definirán algunas clases de complejidad importantes para el tema.

La primera de estas clases es la clase *coNP*. La definición de la clase *coNP*, tomada de[2] es la siguiente:

$$coNP = \{L : \overline{L} \in NP\}.$$

Donde $\overline{L}$ es el complemento de L, definido como $\overline{L} = \{0,1\}^* \setminus L$.

De manera análoga al ejemplo dado para *NP*, para la clase *coNP* se puede encontrar al problema de la tautología como un problema contenido en esta clase de complejidad. El problema de la tautología consiste en decidir si dada una expresión booleana, toda asignación de variables la satisface.

Para mostrar que el problema de la tautología está en *coNP*, basta con notar que el complemento del lenguaje asociado a este problema es el lenguaje de expresiones booleanas donde existe una asignación que no las satisface. Este lenguaje está en *NP*, pues la asignación que no satisface a la expresión es el certificado, y el proceso de verificación es análogo al del problema *SAT*.

Las últimas clases que se mostrarán que dependen del tiempo de ejecución,

son las clases *EXP* y *NEXP*. Estas dos clases serán útiles en secciones posteriores para la discusión sobre las relaciones que hay entre algunas clases de complejidad, en específico, las clases *P* y *NP*.

La definición de la clase *EXP*, tomada de[2] es análoga a la definición de *P*, pero utilizando funciones exponenciales. La definición es la siguiente:

$$EXP = \bigcup_{c \geq 0} DTIME(2^{n^c}).$$

La definición de la clase *NEXP*, tomada de[2], es análoga a la anterior, pero para la clase *NP*. La definición es la siguiente:

$$NEXP = \bigcup_{c \geq 0} NTIME(2^{n^c}).$$

Si bien todas las clases de complejidad mencionadas hasta ahora se clasifican de acuerdo a su tiempo de ejecución, también existen algunas clases de complejidad que se definen por la memoria que requiere una MT o una MTND para computar la solución. Estas clases se denominan clases de computación de espacio delimitado.

Para definir las clases importantes, se comenzará con la definición de las clases generales, al igual que se hizo con las clases de complejidad que dependían del tiempo de ejecución.

La primera de estas clases es la clase *SPACE*. La definición de esta clase, tomada de[2] es la siguiente:

Sea $S : \mathbb{N} \to \mathbb{N}$ una función y $L \in \{0,1\}^*$ un lenguaje. Se dice que $L \in SPACE(S(n))$ si existe una constante c y una MT M que computa a L tal que por cada entrada $x \in \{0,1\}^*$, el número de espacios que son no vacíos en cualquier punto de la ejecución de $M(x)$ es a lo más $c \cdot S(|x|)$.

Análogamente, la clase *NSPACE* se define como la anterior, pero utilizando MTND. La definición de la clase *NSPACE*, tomada de[2] es la siguiente:

Sea $S : \mathbb{N} \to \mathbb{N}$ una función y $L \in \{0,1\}^*$ un lenguaje. Se dice que $L \in NSPACE(S(n))$ si existe una constante c y una MTND M que computa a

L tal que por cada entrada $x \in \{0,1\}^*$, el número de espacios que son no vacíos en cualquier punto de las trayectorias que llegan al estado q_{paro} o $q_{aceptar}$ no deterministas de la ejecución de $M(x)$ es a lo más $c \cdot S(|x|)$.

Finalmente, con las definiciones anteriores se pueden definir las clases de complejidad de computación de espacio delimitado análogas a las clases P y NP, estas clases son la clase $PSPACE$ y $NPSPACE$. La definición de estas dos clases, tomadas de[2] son las siguientes:

$$PSPACE = \bigcup_{c>0} SPACE(n^c)$$

$$NPSPACE = \bigcup_{c>0} NSPACE(n^c)$$

De manera intuitiva, estas clases contienen a los problemas que se pueden resolver utilizando una cantidad de espacio polinomial durante la ejecución, con MT y MTND respectivamente. En el caso de $NSPACE$, la MTND utiliza espacio polinomial en todas sus posibles decisiones.

1.3.2. Relación entre las clases

Como se ha mencionado anteriormente en este trabajo, una de las cuestiones más importantes en la teoría de la complejidad, es la relación entre P y NP. Es bien sabido que $P \subseteq NP \subseteq EXP \subseteq NEXP$, las pruebas pueden encontrarse en[2]. Por otro lado, la pregunta $NP \subseteq P$ es una cuestión que no ha podido resolverse hasta la fecha.

En general, y como se menciona en[2], se cree que $NP \nsubseteq P$ y que por lo tanto $P \neq NP$, pero esto es algo que no se ha podido demostrar, así como tampoco su negación.

Si resultara que $P = NP$ tendría variadas consecuencias en muchas áreas de la computación y las matemáticas. Esto es porque existen algunos problemas de decisión que se encuentran en NP y no se ha encontrado un algoritmo que corra en tiempo polinomial para resolverlos, por lo que no se sabe si están en P.

Las relaciones entre las clases de complejidad mencionadas en la sección anterior podrían ayudar a encontrar la relación que existe entre P y NP. Existen algunos teoremas que implican que si se prueban algunas igualdades o desigualdades de estas clases, implicaría que $P = NP$ o $P \neq NP$.

En[2], se demuestra que si $EXP \neq NEXP$, entonces $P \neq NP$, por lo que bastaría con demostrar lo primero para tener la respuesta sobre la relación entre P y NP. Dado que esta relación aún no está probada, es obvio que lo primero tampoco tiene una prueba, pero al igual que con $P \neq NP$, se conjetura que efectivamente $EXP \neq NEXP$.

Se sabe también que $P \subseteq (NP \cap coNP)$, esto se menciona en[2]. Sin embargo no se sabe si $NP = coNP$. Se menciona además en[2], que si $P = NP$, entonces $P = coNP = NP$, en contrapositiva de esto, se tiene que si $coNP \neq NP$, entonces $P \neq NP$. Al igual que en el caso anterior, no se ha podido demostrar, pero se conjetura que en efecto $coNP \neq NP$.

Por otro lado, se sabe que $P \subseteq NP \subseteq PSPACE \subseteq NPSPACE$ y además $PSPACE = NPSPACE$, la demostración de esto puede verse en[2]. Sin embargo, no se sabe si $P = PSPACE$, o en cambio si $NP = PSPACE$, aunque se conjetura que esto no es así. Por contención de conjuntos, si $P = PSPACE$, entonces $P = NP = PSPACE = NPSPACE$.

Si bien estas afirmaciones indican que el conocer la relación entre dos clases de complejidad implica encontrar la relación entre otras, no da una manera de buscar ninguna de estas relaciones para empezar.

En la siguiente sección se mostrará una manera de clasificar problemas que son representativos de algunas clases de complejidad, de tal manera que si se demostrara que el problema pertenece a una clase menor, se podría probar la igualdad entre ambas clases.

1.3.3. Reducciones, dureza y completitud.

Para caracterizar a las clases de complejidad por los problemas que son más difíciles de resolver en esta clase, se definirá el término de dureza computacional. De manera intuitiva, la dureza computacional se define como ser al menos tan difícil de resolver como cualquier problema de cierta

clase de complejidad. De esta forma, un problema *NP-Duro* es al menos tan difícil de resolver como cualquier problema en *NP*.

Para definir formalmente la dureza computacional, es necesario definir el término reducible en tiempo polinomial. Esta definición, tomada de[2], es la siguiente:

Se dice que un lenguaje $A \subseteq \{0,1\}^*$ es Karp reducible en tiempo polinomial a un lenguaje $B \subseteq \{0,1\}^*$ (algunas veces sólo llamado "reducible en tiempo polinomial") denotado como $A \leq_p B$ si existe una función computable en tiempo polinomial $f : \{0,1\}^* \to \{0,1\}^*$ tal que para toda $x \in \{0,1\}^*$, $x \in A \Leftrightarrow f(x) \in B$.

Una propiedad importante de las reducciones polinomiales es que son transitivas, es decir, si $A \leq_p B$ y $B \leq_p C$, entonces $A \leq_p C$. Esta propiedad es fácilmente verificable, pues basta con hacer una composición de funciones polinomiales para llegar de A a C, y la composición de dos funciones polinomiales es otra función polinomial.

Una vez definido el término de reducible en tiempo polinomial, se puede definir la dureza computacional. Para más practicidad, se definirá el término *NP-Duro*. Los términos *coNP-Duro*, *EXP-Duro*, *NEXP-Duro*, y *PSPACE-Duro*, se definen de manera análoga. La definición de *NP-Duro*, obtenida de[2], es la siguiente:

Se dice que B es *NP-Duro* si $A \leq_p B$, $\forall A \in NP$.

Las definiciones análogas para *coNP-Duro*, *EXP-Duro*, *NEXP-Duro*, y *PSPACE-Duro*, se encuentran también en[2].

Por otro lado, otra definición importante es la definición de completitud. Las definiciones de *NP-Completo*, *coNP-Completo*, *EXP-Completo*, *NEXP-Completo*, y *PSPACE-Completo* se encuentran en[2]. Al igual que con las definiciones anteriores, se definirá el término *NP-Completo*, y los demás son análogos a este.

Se dice que A es *NP-Completo* si A es *NP-Duro* y $A \in NP$.

Por la propiedad de transitividad de las reducciones polinomiales, para demostrar que un lenguaje B es *NP-Duro*, basta con encontrar una reducción

polinomial de algún lenguaje A que se sabe que es *NP-Duro* o *NP-Completo* a B.

La importancia de encontrar problemas que sean *NP-Duros* y *NP-Completos*, puede verse en los siguientes teoremas, los cuales se mencionan en[2]:

1. Si un lenguaje A es *NP-Duro* y $A \in P$, entonces $P = NP$.

2. Si un lenguaje A es *NP-Completo* entonces $A \in P \Leftrightarrow P = NP$.

Como se puede notar, bastaría con que se muestre que algún problema *NP-Duro* o *NP-Completo* esté en P para resolver la cuestión sobre la relación entre P y NP.

El primer problema *NP-Completo* fue mostrado en 1971 en[6]. Este problema es llamado problema de satisfacibilidad. Dentro de[6] se demuestra que todo lenguaje en NP se puede reducir polinomialmente al problema SAT.

En 1972, Karp mostró una lista, ahora muy bien conocida, de 21 problemas *NP-Completos* en[15].

Análogamente, como se menciona en[2], si se prueba que un problema *PSPACE-Duro* o *PSPACE-Completo* está en P o NP, se demostraría que $PSPACE = P$ o $PSPACE = NP$ respectivamente.

De igual manera que con los problemas *NP-Duros*, para demostrar que un lenguaje B es *PSPACE-Duro*, basta con encontrar una reducción polinomial de algún lenguaje que se sabe que es *PSPACE-Duro* o *PSPACE-Completo* A al lenguaje B.

1.4. Problemas de decisión representativos en videojuegos

Existen distintos géneros de videojuegos con distintos objetivos. Los videojuegos que se analizarán en este trabajo se encuentran entre los géneros

rompecabezas, carreras, plataformas en 2D, plataformas en 3D y exploración de calabozos en vista aérea.

Algunos videojuegos del género rompecabezas tienen como objetivo destruir una cantidad de bloques específicos, o destruir todos los que están en pantalla. Este objetivo puede tener limitantes como número de acciones permitidas o límite de tiempo. En general, si se tiene un límite de tiempo, se considera que se puede realizar un número limitado de movimientos en ese tiempo, por lo que se reduce a tener un límite de movimientos permitidos.

En algunos videojuegos de rompecabezas se tiene como problema de decisión el decidir si es posible destruir una determinada cantidad de elementos con una secuencia de movimientos legales, decidir si se puede destruir un elemento específico con una secuencia de movimientos legales, decidir si se pueden destruir todos los elementos con una cantidad máxima dada de movimientos legales, o decidir si se puede destruir un elemento específico con una cantidad máxima dada de movimientos legales.

Los videojuegos de rompecabezas generalmente son fáciles de analizar, pues cada movimiento se toma como un paso en la computación de la MT. Algunos problemas en videojuegos de este género analizados anteriormente pueden encontrarse en [5] y [27].

Los videojuegos de carreras tienen objetivos como llegar a la meta en la menor cantidad de tiempo o llegar antes que un contrincante. En algunos videojuegos, y en particular en el que se analiza en este trabajo, se tienen objetos que al tocarlos, el vehículo se detiene durante unos momentos, en este caso es necesario considerar que el tiempo que te quita tomar uno de estos objetos es también fijo.

En algunos videojuegos de carreras se pueden destruir los objetos del escenario tomando ciertos objetos o eligiendo un camino específico. Este tipo de videojuegos admite modos multijugador.

Para analizar videojuegos de carreras, se asumen ciertas cosas para poder considerarlos problemas de decisión, como que existe un tiempo óptimo para recorrer una sección del escenario, de tal manera que decidir si tomar uno u otro camino toma cantidades de tiempo deterministas.

Los problemas de decisión que se pueden encontrar en algunos videojuegos

de carreras son decidir si dado un escenario, en el cual se conoce el tiempo óptimo para recorrer las distintas bifurcaciones del mismo, puede completarse en un tiempo dado, o si dado un escenario en el cual se juegan dos jugadores, existe una serie de decisiones que le garantice a alguno de los dos jugadores siempre ganar la carrera. Algunos problemas en videojuegos de este género analizados anteriormente pueden encontrarse en[4].

Finalmente, los videojuegos de plataformas 2D, plataformas 3D y exploración de calabozos en vista aérea, tienen en general objetivos similares. Uno de los objetivos más populares es llegar al final de un nivel, que puede reducirse a la idea de llegar desde el punto A al punto B. Algunos videojuegos de estos géneros tienen como objetivo el recolectar un conjunto o cantidad determinada de objetos, esparcidos a lo largo del escenario. En general, en este tipo de videojuegos se tienen enemigos con movimientos determinados, o que dependen de las acciones del jugador.

Para analizar videojuegos de plataformas 2D, plataformas 3D y exploración de calabozos en vista aérea, se necesita determinar las acciones que son posibles para el jugador dado el estado del escenario en un momento dado, de tal manera que el problema consista en determinar el resultado final dependiendo de una serie de decisiones del jugador.

Los problemas de decisión que se pueden encontrar en algunos videojuegos de plataformas 2D, plataformas 3D y exploración de calabozos en vista aérea, son decidir si dado un escenario, es posible llegar del punto A al punto B siguiendo las reglas del videojuego, decidir si dado un escenario con objetos, es posible obtener una cantidad mínima de objetos, o si dado un escenario, es posible terminarlo en una cantidad de tiempo dado. Algunos problemas en videojuegos de este género analizados anteriormente pueden encontrarse en[27],[1],[8] y[9].

En general, la mayoría de los videojuegos cuentan con ciertas limitantes que dependen del hardware para el que fueron construidos, como limitaciones de memoria que obligan a que los escenarios tengan un tamaño finito, o un límite de objetos o enemigos que obligan a que algunos de estos desaparezcan o regresen a su estado inicial.

En general, para analizar los problemas en videojuegos se toma una versión general del mismo que no cuenta con estas limitantes. También se suelen omitir algunas reglas o elementos de ciertos videojuegos, para de esta forma indicar si su complejidad depende de estos elementos o es independiente

de estos. En caso de que uno de estos elementos se omita, se especificará el elemento omitido y se considerará un problema distinto o una variante del problema original.

1.5. Resumen

En este capítulo se presentan los fundamentos teóricos esenciales de este trabajo. Se explica de manera detallada las definiciones necesarias sobre Máquinas de Turing, complejidad computacional y clases de complejidad.

Sobre las clases de complejidad, se da una definición de las más conocidas e importantes en el área, como son las clases P, NP, $PSPACE$, $coNP$, $NPSPACE$, EXP, $NEXP$, entre otras.

Una vez definidas las clases, se mencionan las relaciones entre estas clases que son conocidas y estudiadas en el área. Se hace un énfasis en la importancia de la relación que existe entre las clases P y NP y la conjetura que se tiene de que $P \neq NP$.

Posteriormente se definen los términos dureza computacional, completitud computacional y reducción polinomial. A partir de estas definiciones, se enfatiza la importancia de buscar nuevos problemas *NP-Duros* dado que encontrar un problema *NP-Duro* que esté en P, demostraría que $P = NP$.

Finalmente, se mencionan varios problemas de decisión estudiados dentro de distintos videojuegos. Estos problemas se ejemplifican por género del videojuego, pues en general, para los videojuegos de un mismo género se estudian problemas de decisión similares.

2 Estado del arte

En esta sección se presenta una variedad de videojuegos cuya complejidad computacional ha sido estudiada en trabajos previos a este. Para tener un mejor orden en las demostraciones de dureza de estos videojuegos, se presentarán primero las reglas generales de los videojuegos estudiados. Posteriormente se presentan técnicas generales para el estudio de la complejidad en videojuegos.

Finalmente se presentan las demostraciones de *NP-Dureza* y *PSPACE-Completitud* publicadas en otros trabajos. Los videojuegos estudiados en esta sección son Super Mario Bros., The Legend of Zelda: A Link To The Past, Donkey Kong Country, Pokémon Rojo/Azul, Metroid, Mario Kart 64, Portal y Tetris.

2.1. Videojuegos previamente investigados y sus reglas

En esta sección se presentarán las características, reglas y elementos de algunos videojuegos estudiados en trabajos previos a este. Se presentarán solamente los elementos que influyen en el desarrollo de las demostraciones. Además, algunos de estos elementos se generalizan para poder tener una versión del videojuego que pueda expresar instancias con dureza computacional.

Dado que todos los videojuegos estudiados en esta sección tienen limitaciones de espacio en su versión original, para las versiones generalizadas se supondrá que algunos elementos son no acotados. Estos elementos se especificarán en cada videojuego.

Si bien en algunos de estos videojuegos pueden existir ciertos errores que permiten romper algunas reglas de los mismos (estos errores son generalmente conocidos como *"glitches"*). Para estas versiones generalizadas se

supondrá que estos errores no existen, pues se considera la versión del videojuego que fue pensada por los desarrolladores.

2.1.1. Super Mario Bros.

Super Mario Bros. es un videojuego de plataformas en 2D desarrollado originalmente para la consola Nintendo Entertainment System en 1985. En este videojuego, el personaje principal Mario debe llegar hasta el final de una serie de niveles para rescatar a la princesa Toadstool del villano Bowser al final del último nivel[21]. En la figura 2.1 puede observarse una imagen del videojuego original en ejecución.

Figura 2.1: Super Mario Bros. en ejecución.

Mario puede caminar o correr a la izquierda o a la derecha a una velocidad determinada. Además, puede saltar una altura y distancia determinada. Los saltos de Mario le permiten subir a plataformas que se encuentren a mayor de elevación con respecto a él, y puede saltar por encima de precipicios. Si al saltar, Mario golpea el techo por debajo, su salto se verá obstruido y caerá. Mario puede moverse a la izquierda o a la derecha mientras cae.

Mario iniciará siendo pequeño, de un cuadro de altura. Si Mario es golpeado por un enemigo en estado pequeño, morirá. Mario inicia el videojuego con tres vidas. Cuando Mario se queda sin vidas, perderá.

En los niveles puede existir un punto de control. Este punto de control no es visible para el jugador. Si Mario camina más allá del punto de control y muere, reaparecerá en este punto de control

En cada nivel se tiene un límite de tiempo, este límite puede variar dependiendo del nivel. En el videojuego original se utilizan múltiplos de 100 para el límite de tiempo de los niveles, esto no influye en la demostración pero esta característica se conservará para tener mayor fidelidad con el videojuego original. Cuando el límite de tiempo se termina, Mario morirá. Al reaparecer en un punto de control, el límite de tiempo puede ser diferente al límite inicial del nivel.

En los escenarios pueden encontrarse monedas que Mario puede recoger al tocarlas. Mario inicia con un contador de monedas de 0. En el videojuego original, cuando el contador de monedas de Mario llega a 100, recibirá una vida y el contador se reiniciará. En la figura 2.2 puede observarse un nivel de Super Mario Bros. con monedas.

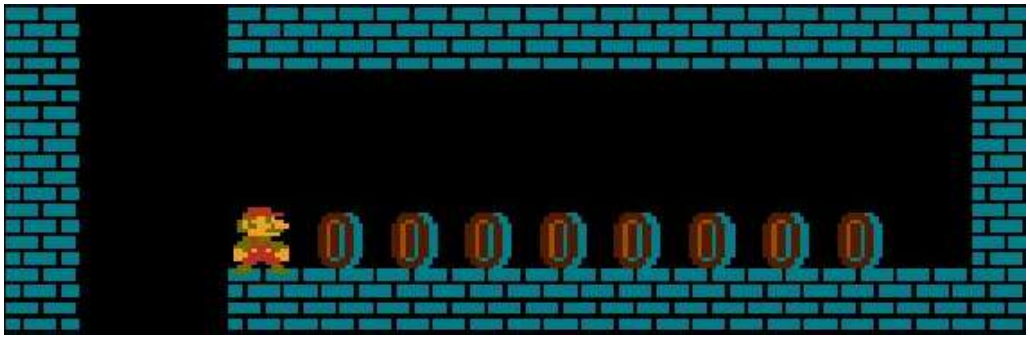


Figura 2.2: Monedas en Super Mario Bros.

En los escenarios pueden encontrarse tuberías. Estas tuberías salen del fondo del nivel y pueden encontrarse al nivel del suelo o más elevadas. Mario puede caminar o correr sobre la tubería como si fuera una plataforma. Mario puede entrar a las tuberías desde arriba presionando un botón. Cuando Mario entra a una tubería saldrá por otra tubería en alguna parte del nivel. Algunas tuberías pueden servir sólo para salir de ellas, y no se

puede entrar de vuelta. En la figura 2.3 se observan estas tuberías y su funcionamiento.

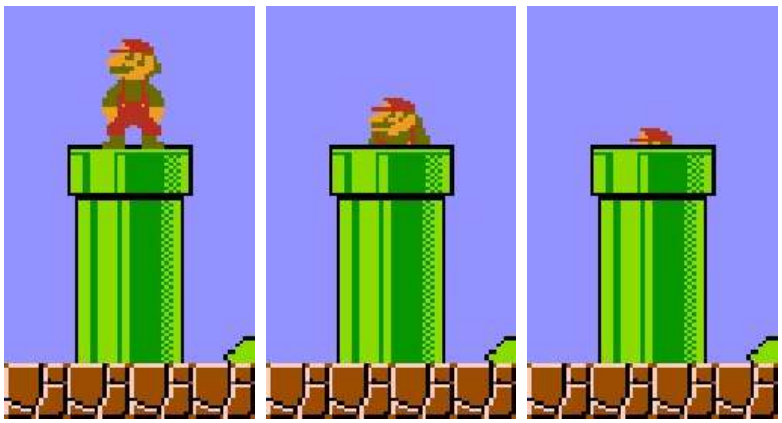


Figura 2.3: Tuberías en Super Mario Bros.

Existe un objeto llamado super champiñón. Al tocar un super champiñón, Mario crecerá a dos cuadros de altura. Si Mario es golpeado por un enemigo mientras está en este estado, volverá al estado de Mario pequeño. En la figura 2.4 se observa el super champiñón y su funcionamiento.

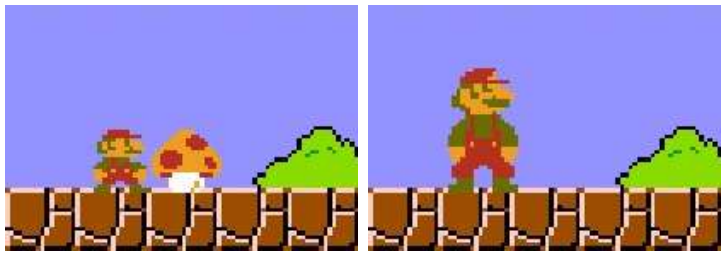


Figura 2.4: Super champiñón en Super Mario Bros.

Existe otro objeto llamado estrella. Si Mario toca a una estrella se volverá invulnerable durante un periodo limitado de tiempo. En la figura 2.5 se observa la estrella y su funcionamiento.

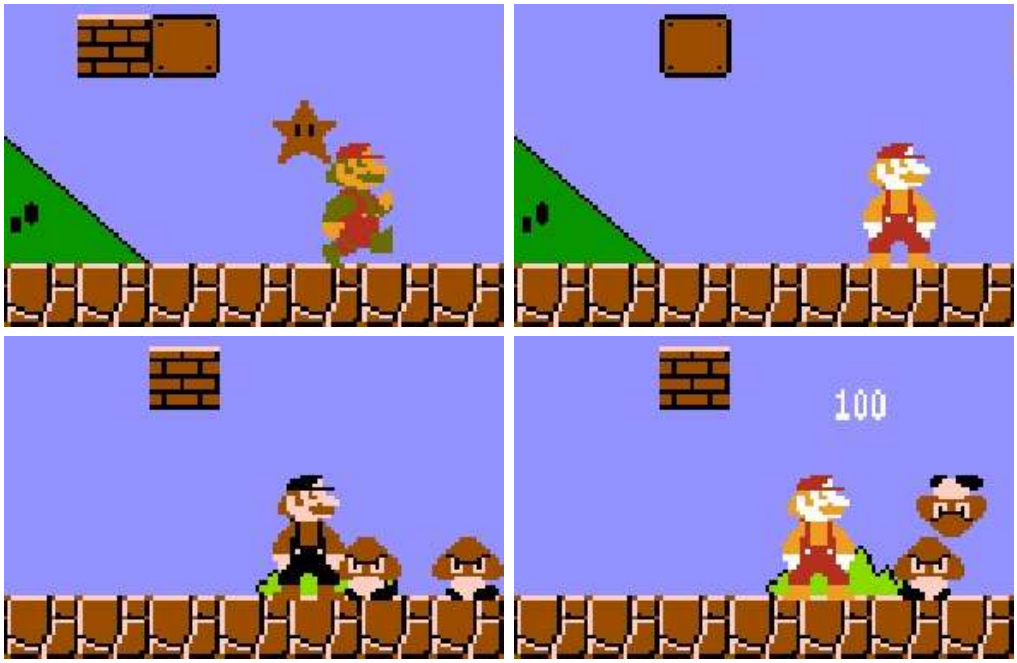


Figura 2.5: Estrella en Super Mario Bros.

Cuando Mario se encuentre en estado grande, no podrá atravesar espacios de un cuadro en los que no cabe. Si Mario se encuentra en estado grande, puede agacharse. Si Mario se agacha después de correr a través de varios cuadros, la inercia hará que avance aproximadamente un cuadro mientras está agachado. Un ejemplo de un pasillo estrecho se observa en la figura 2.6.

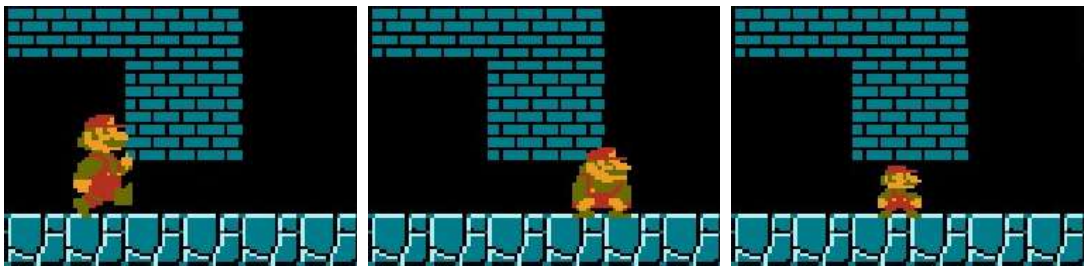


Figura 2.6: Pasillo estrecho en Super Mario Bros.

En los escenarios pueden encontrarse, además del suelo común, varios tipos de bloques. Para las demostraciones presentadas se utilizarán tres tipos de bloques, bloques simples, bloques de objeto y bloques de ladrillo. Mario puede caminar por encima de todos los bloques y no puede atravesarlos.

Los bloques simples son indestructibles e inamovibles. Existen dos formas de este bloque, pero a nivel de jugabilidad no tienen ninguna diferencia.

Los bloques de objeto tienen un signo de interrogación en el centro. Estos bloques pueden ser activados por Mario saltando debajo de ellos y golpeandolos por debajo. Al activarlo, el bloque de objeto se transformará en un bloque simple y soltará un objeto. El objeto que soltará está predefinido, pero puede ser una estrella o un super champiñón.

Los bloques de ladrillo pueden ser destruidos por Mario grande si este salta por debajo de ellos y los toca. Si Mario se encuentra en estado pequeño, no podrá destruir los bloques de ladrillo, en lugar de esto hará que el bloque se levante medio cuadro. Si un objeto o enemigo se encuentra en el suelo por encima de un bloque de ladrillo, y Mario golpea ese bloque, el objeto o enemigo dará un brinco de un cuadro, mientras continúa con su movimiento. En la figura 2.7 se observan los distintos tipos de bloques y su funcionamiento.

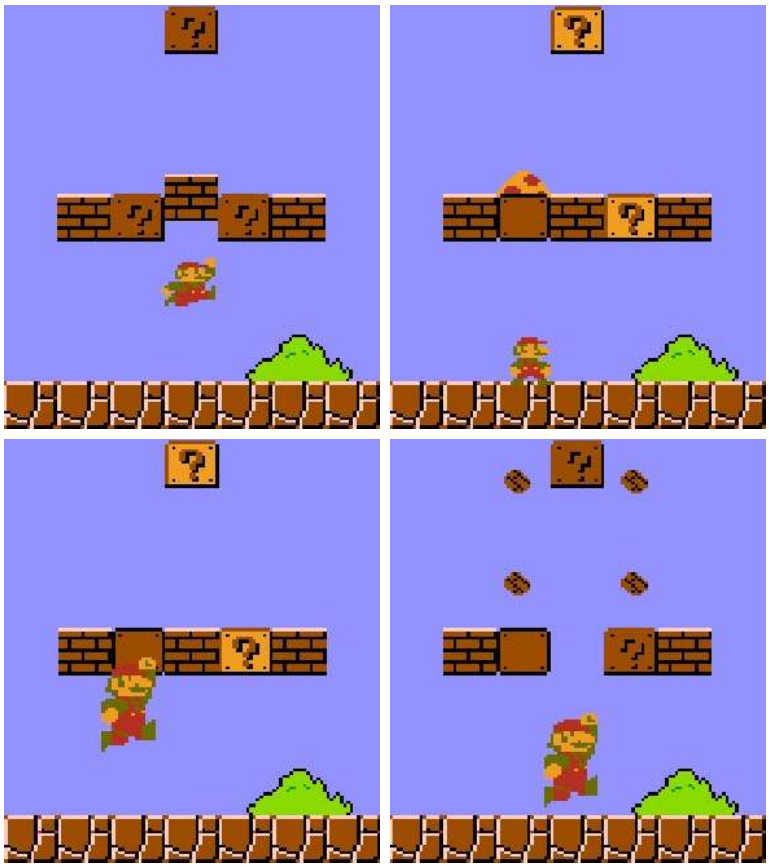


Figura 2.7: Bloques en Super Mario Bros.

Cuando una estrella es liberada de un bloque de objeto, saltará por encima del bloque hacia la derecha, al tocar el suelo saltará de nuevo infinitamente, y al tocar una pared rebotará al lado contrario. Cuando un super champiñón es liberado de un bloque de objeto, se moverá por encima del bloque hacia la derecha, si choca con una pared rebotará al lado contrario. Si un super champiñón llega al borde de una plataforma elevada, caerá por ella.

Existen varios tipos de enemigos utilizados en las demostraciones de este trabajo. El primer enemigo a destacar es el llamado goomba. Los goomba se mueven de la misma forma que un super champiñón, pero si Mario los toca por un lado, o por debajo, sufrirá daño. Mario puede eliminar un goomba si cae encima de él. En la figura 2.8a puede observarse un goomba.

Otro enemigo utilizado son las barras de fuego. Las barras de fuego son una línea de esferas de fuego, que giran alrededor de su eje. Este eje se encuentra en un extremo de la línea de esferas de fuego. Las barras de fuego se encuentran flotando en el aire y pueden atravesar bloques. Al estar en el aire, los golpes inferiores de bloques de ladrillo no los afectan. Las barras de fuego pueden tener tamaño variable y no pueden ser destruidas. Si Mario toca una esfera de fuego sufrirá daño. En la figura 2.8b puede observarse una barra de fuego.

El último enemigo que se utilizará es el spiny. Este enemigo tiene un movimiento igual al de un goomba. A diferencia de los goombas, si Mario trata de destruir a un spiny cayendo encima de él, Mario recibirá daño y el spiny no. En la figura 2.8c se observa un spiny.

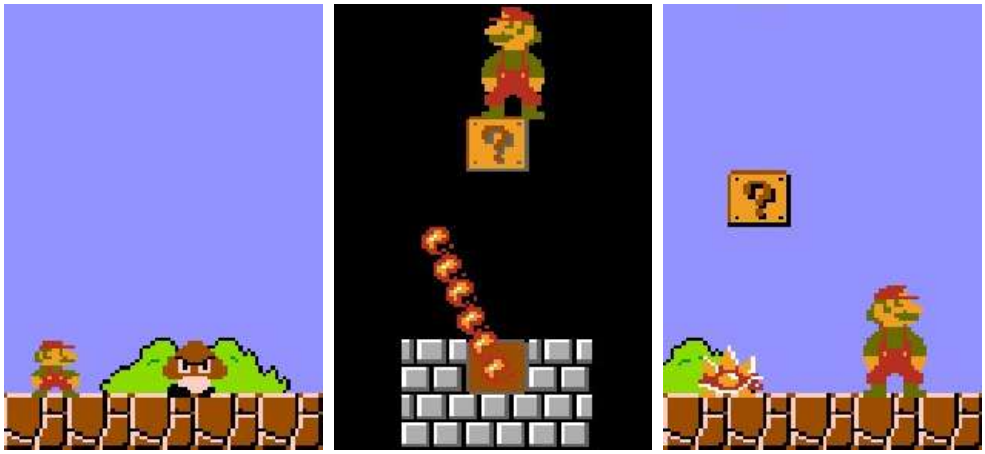


Figura 2.8: Enemigos en Super Mario Bros., goomba, barra de fuego y spiny.

En el videojuego original, se tiene un límite de objetos que pueden verse en pantalla, y en general, la altura está limitada por la que puede verse en una sóla pantalla. Además, en el videojuego original se avanza hacia la derecha, y la pantalla avanza junto con Mario. Cuando la pantalla se ha movido, no es posible regresar hacia la izquierda. Para algunas demostraciones se utilizará un tamaño no acotado para este escenario y pantalla, por lo que se supondrá que la limitación de altura y no poder avanzar a la izquierda no existen.

Otras limitaciones que existen son algunas referentes a los objetos. en el videojuego original hay un límite de objetos que pueden estar activos al mismo tiempo. En esta versión generalizada se supondrá que el número de objetos que pueden estar activos es no acotado. Otra limitación es que los objetos desaparecen al salir de la pantalla. Como se está suponiendo que el tamaño de la pantalla es no acotado, este problema no se encuentra en esta versión del videojuego.

Para mayor información del videojuego y sus reglas, consultar[21].

2.1.2. Donkey Kong Country

Donkey Kong Country es un videojuego de plataformas en 2D desarrollado originalmente para la consola Super Nintendo Entertainment System en

1994. En este videojuego, los personajes principales Donkey Kong y Diddy Kong deben llegar hasta el final de una serie de niveles para recuperar los plátanos, robadas por el villano King K. Rool, al final del último nivel[10]. En la figura 2.9 puede observarse una imagen del videojuego original en ejecución.

Figura 2.9: Donkey Kong Country en ejecución.

En el videojuego controlas a uno de los dos Kong. Inicialmente sólo se controla a Donkey Kong. Por simplicidad, y dado que no hay una diferencia importante en jugabilidad entre ambos personajes, se describirán las reglas del videojuego refiriéndose completamente a Donkey Kong cuando se hable de una acción realizable por cualquiera de los dos personajes.

Donkey Kong puede caminar o correr a la izquierda o a la derecha a una velocidad determinada. Además, puede saltar una altura y distancia determinada. Los saltos de Donkey Kong le permiten subir a plataformas que se encuentren a mayor elevación con respecto a él, y puede saltar por encima de precipicios. Si al saltar, Donkey Kong golpea el techo por debajo, su salto se verá obstruido y caerá. Donkey Kong puede moverse a la izquierda o a la derecha mientras cae.

En los escenarios puede haber precipicios, por los cuales si Donkey Kong cae, el jugador perderá a ambos Kong y perderá una vida.

En los escenarios pueden encontrarse barriles en el suelo. Donkey Kong puede tomar los barriles y lanzarlos a su izquierda o derecha. Una vez lanzado, el barril continuará rodando hasta chocar con una pared, un enemigo o caer a un precipicio. Si un barril cae de una plataforma elevada, se moverá en diagonal hacia abajo y la dirección en la que se movía antes de caer. Cuando un barril choca con una pared o enemigo, el barril será destruido. En la figura 2.10 se observa un barril siendo lanzado.

Figura 2.10: Barriles en Donkey Kong Country.

En los escenarios puedes encontrar barriles DK. Al tocarlos, los barriles DK añaden a la reserva al Kong que el jugador no esté controlando, en caso de que sólo tengas a uno. Si el jugador cuenta con ambos Kong, el barril DK no le otorgará nada al jugador. Cuando Diddy Kong se encuentra en la reserva, el jugador puede cambiarlo para jugar con él en cualquier momento, poniendo a Donkey Kong en la reserva. En la figura 2.11 se observa un barril DK.

Figura 2.11: Barriles DK en Donkey Kong Country.

En los escenarios pueden encontrarse cuerdas flotantes. Donkey Kong puede adherirse a estas cuerdas y subir o bajar por ellas a una velocidad determinada. También puede saltar a la izquierda o derecha de la cuerda estando a cualquier altura de la misma. En la figura 2.12 se observa una cuerda flotante.

Figura 2.12: Cuerdas flotantes en Donkey Kong Country.

En los escenarios pueden encontrarse cañones de barril. Donkey Kong puede entrar en estos cañones para ser lanzado por ellos en una dirección dada a una distancia predeterminada para ese cañón, pero que puede cambiar de un cañón a otro. Estos cañones pueden encontrarse a nivel del suelo o flotando en el escenario.

Existen dos tipos de estos cañones, los cañones manuales y los automáticos. La diferencia entre estos es que en el primero el jugador decide cuándo lanzar a Donkey Kong, mientras que en el segundo se lanza inmediatamente al entrar. Los cañones automáticos tienen una dirección predeterminada a la que se lanzará Donkey Kong.

Por otro lado, los cañones manuales pueden tener una dirección predeterminada o encontrarse girando. Si el cañón se encuentra girando, el jugador puede elegir cuándo detenerlo para que Donkey Kong sea lanzado en la dirección en la que se encuentra el cañón al detenerse. En la figura 2.13 se observa un cañón de barril y su funcionamiento.

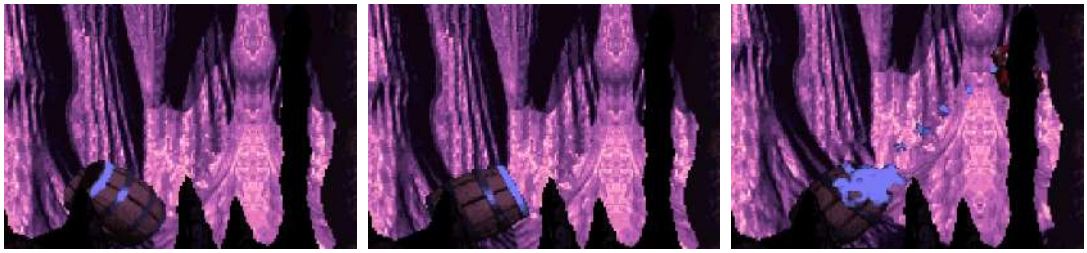


Figura 2.13: Cañones de barril en Donkey Kong Country.

En los escenarios pueden encontrarse llantas. Las llantas pueden ser empujadas por Donkey Kong, hacia la izquierda o la derecha. Si bien en el videojuego original las llantas regresan a su posición inicial al salir de pantalla, en esta versión generalizada se supondrá que su posición es persistente.

Algunos escenarios tienen suelo congelado. Si una llanta es empujada cuesta abajo por una superficie con suelo congelado inclinada, la llanta continuará rodando hasta bajar toda la superficie inclinada, y avanzará un poco más allá de la misma, dependiendo de la inclinación y la longitud de la superficie inclinada. Si Donkey Kong cae encima de una llanta, rebotará hacia arriba. En la figura 2.14 se observa una llanta y su funcionamiento.

Figura 2.14: Llantas en Donkey Kong Country.

En los escenarios pueden encontrarse varios tipos de enemigos. En esta versión generalizada se utilizará solamente un tipo de enemigo, los zingers. Los zingers son enemigos parecidos a avispas gigantes. Los zingers pueden volar, por lo que pueden estar posicionados en cualquier parte del escenario, no necesariamente a nivel del suelo. Si Donkey Kong toca a un zinger, Donkey Kong morirá.

Los zingers pueden tener un patrón de movimiento de izquierda a derecha,

de arriba hacia abajo, o quedarse estático. Si un barril golpea a un zinger, morirá. En la figura 2.15 se observa un zinger siendo eliminado.

Figura 2.15: Zinger siendo eliminado por un barril en Donkey Kong Country.

Si Donkey Kong muere y tiene a Diddy Kong en la reserva, Donkey Kong se perderá pero se podrá seguir jugando con Diddy Kong. En esta situación, cuando Donkey Kong muere, Diddy aparecerá con un periodo de invencibilidad limitado, pero con el suficiente tiempo para atravesar al enemigo que mató a Donkey Kong. Si Donkey Kong muere y no se tiene reserva, perderá una vida.

Para mayor información del videojuego y sus reglas, consultar [10].

2.1.3. The Legend of Zelda: A Link to the Past

The Legend of Zelda: A Link to the Past es un videojuego de acción y aventuras en vista aérea desarrollado para la consola Super Nintendo Entertainment System en 1991 [28]. En este videojuego, el personaje principal Link debe atravesar una serie de mazmorras, conformadas de múltiples cuartos, para rescatar a la princesa Zelda derrotando al villano Ganondorf al final de la última mazmorra. En la figura 2.16 puede observarse una imagen del videojuego original en ejecución.

Figura 2.16: The Legend of Zelda: A Link to the Past en ejecución.

Link puede moverse en ocho direcciones, arriba, abajo, izquierda, derecha y diagonales a una velocidad determinada. Además, Link cuenta con un gancho que puede lanzar en cuatro direcciones, arriba, abajo, izquierda y derecha. El gancho tiene una distancia máxima definida. Cuando el gancho golpea ciertos elementos dados del escenario, se adherirá a estos y Link será atraído hacia el elemento, quedando al lado del mismo.

En los escenarios pueden encontrarse cofres. En el videojuego original, estos cofres contienen objetos, pero no se utilizará en esta versión generalizada. Los cofres no pueden moverse de su lugar. El gancho puede adherirse a los cofres. Link no puede atravesar los cofres.

En los escenarios puede haber vacío. Si Link trata de caminar normalmente por este vacío caerá, sufrirá daño y regresará a la entrada de la habitación donde tocó el vacío. El gancho puede atravesar el vacío. Si Link utiliza el gancho para adherirse a un elemento que esté del otro lado del vació, podrá atravesarlo y llegar al elemento adherible. En la figura 2.17 se observa a Link atravesando el vacío con ayuda del gancho.

Figura 2.17: Link adhiriendo el gancho a un cofre para atravesar un vacío en The Legend of Zelda: A Link to the Past.

En los escenarios pueden encontrarse paredes que dividen al mismo en secciones. Ni Link ni el gancho pueden atravesar estas paredes.

En los escenarios pueden encontrarse plataformas a distintas elevaciones. Las plataformas a distinta elevación pueden pasar una por debajo de otra. En la figura 2.18 se observan plataformas a distinta elevación, con una pasando debajo de otra.

Figura 2.18: Plataforma que pasa por debajo de otra a mayor elevación en The Legend of Zelda: A Link to the Past.

En los escenarios pueden haber escaleras que te permitan subir y bajar a plataformas de distinta de elevación. Si no hay una pared entre dos plataformas a distinta elevación, Link podrá bajar a la plataforma inferior desde la superior, pero no podrá subir de la inferior a la superior.

Por limitaciones del videojuego, cuando hay dos plataformas juntas a distinta elevación, tanto si hay escalera como si no hay, parecerá que se encuentran separados por varios cuadros. Estos cuadros corresponden al lateral de la superficie entre la plataforma inferior y superior. En la figura 2.19 se ejemplifican las distintas formas de subir y bajar de una plataforma a otra.

Figura 2.19: Plataformas a distinta elevación en The Legend of Zelda: A Link to the Past. Se puede bajar de la de mayor altura a la de menor por un borde sin pared, pero no se puede subir. Por las escaleras se puede subir y bajar.

En los escenarios pueden encontrarse bloques de metal. Link puede empujar estos bloques cuando se encuentra al lado de ellos un cuadro hacia una dirección dada. Una vez que el bloque se haya movido, no puede regresar a su posición original. Un bloque no puede ser empujado a través de una pared. El gancho puede adherirse a los bloques. Link no puede atravesar los bloques. En la figura 2.20 se observan bloques y su funcionamiento.

Figura 2.20: Bloques en The Legend of Zelda: A Link to the Past. Los bloques pueden ser empujados una vez pero no pueden volver a su posición inicial.

En los escenarios pueden encontrarse interruptores a nivel del suelo. Cada interruptor está ligado a una o varias puertas, y al pisarlo cambiará el estado de las puertas entre abierto y cerrado. Las puertas siempre inician en estado cerrado. En esta versión generalizada se supondrá que el estado de abierto o cerrado es persistente, incluso al salir de un cuarto y volver a entrar. En la figura 2.21 se observa un interruptor y una puerta.

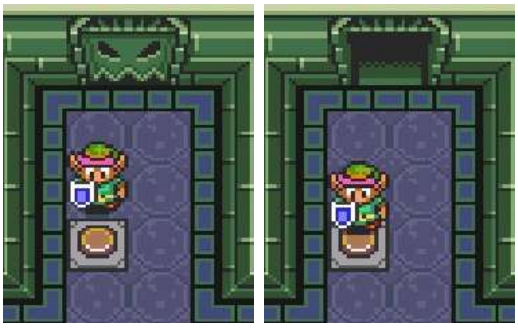


Figura 2.21: Interruptores y puertas en The Legend of Zelda: A Link to the Past.

En los escenarios pueden encontrarse pilares en el suelo. Estos pilares tienen un estado inicial que puede ser arriba o abajo. En estado de abajo, los pilares se comportan como suelo normal, por lo que Link puede andar encima de ellos sin problema. En estado de arriba, los pilares funcionan como una pared, por lo que impiden que Link atraviese al otro lado de ellos.

En el escenario puede haber un cristal. Este cristal puede ser activado por Link una cantidad de veces no acotada cuando se encuentra al lado de él. Al activar el cristal, todos los pilares cambiarán de estado. Al igual que

con las puertas, en esta versión generalizada se supondrá que el estado de arriba o abajo de todos los pilares es persistente, incluso al salir de un cuarto y volver a entrar. En la figura 2.22 se observa un cristal cambiando el estado de un grupo de pilares.

Figura 2.22: Cristal y pilares en The Legend of Zelda: A Link to the Past.

En los escenarios se pueden encontrar teletransportadores. Estos teletransportadores tienen una salida predeterminada. Cuando Link camina encima de un teletransportador, es teletransportado a la salida del mismo automáticamente. En la figura 2.23 se observa un teletransportador y su funcionamiento.

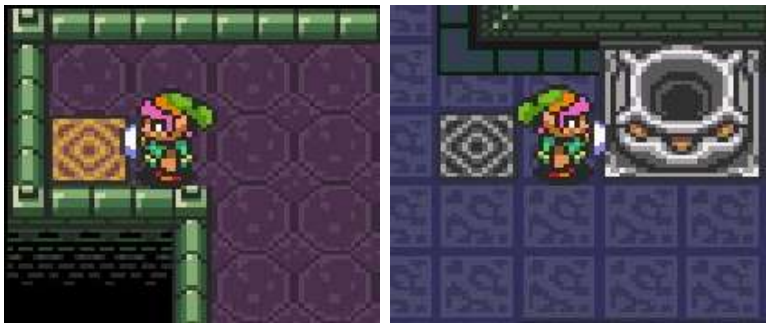


Figura 2.23: Entrada y salida de un teletransportador en The Legend of Zelda: A Link to the Past.

Para mayor información del videojuego y sus reglas, consultar [28].

2.1.4. Metroid

Metroid es un videojuego de plataformas y disparos en 2D desarrollado originalmente para la consola Famicom Disk System en 1986. En este videojuego, la cazadora espacial Samus debe explorar el planeta Zebes, con la finalidad de erradicar a una especie alienígena llamada Metroid [18].

A diferencia de otros videojuegos de plataformas estudiados, Metroid no se compone de niveles, si no de un gran escenario para explorar dividido en distintas habitaciones. En la figura 2.24 puede observarse una imagen del videojuego original en ejecución.

Figura 2.24: Metroid en ejecución.

Samus puede caminar a la izquierda o derecha a una velocidad determinada. Además, puede saltar a una distancia y altura determinadas. Los saltos de Samus le permiten subir a plataformas que se encuentren a mayor elevación con respecto a ella, y puede saltar por encima de precipicios. Si al saltar, Samus golpea el techo por debajo, su salto se verá obstruido y caerá. Samus puede moverse a la izquierda o a la derecha mientras cae.

Samus mide dos cuadros de altura. Samus tiene una habilidad que le permite transformarse en una esfera cuando se encuentre a nivel del suelo. Estado en estado de esfera, Samus mide un cuadro de altura. En estado de esfera, Samus no puede saltar, pero puede moverse de izquierda a derecha y caer por orillas de plataformas altas. Si Samus se encuentra saltando o cayendo, no puede convertirse en esfera.

En los escenarios pueden encontrarse espacios de altura de un cuadro, por los que Samus sólo puede pasar en estado de esfera. Samus puede salir del estado de esfera si se encuentra a nivel del suelo en un espacio de dos o más cuadros de altura. Si Samus se encuentra en el aire o en un espacio de un cuadro de altura, no podrá salir del estado de esfera. En la figura 2.25 se observa a Samus convirtiéndose en esfera y pasando por un espacio angosto.

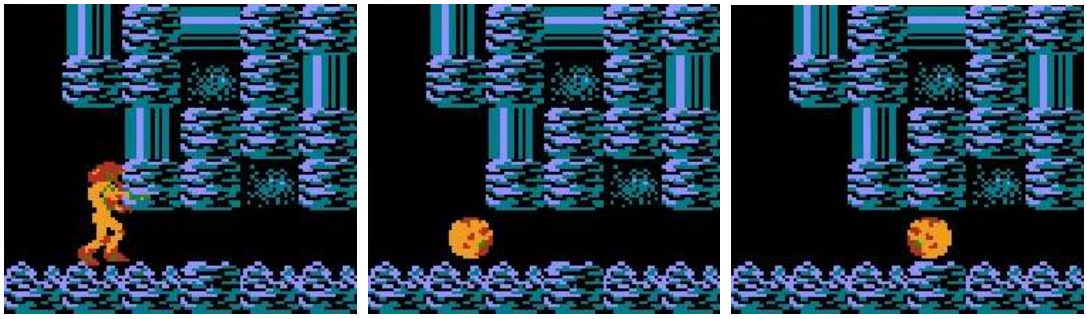


Figura 2.25: Samus convirtiéndose en esfera para entrar en un pasillo angosto en Metroid.

Samus puede disparar hacia arriba, izquierda o derecha, pero no hacia abajo. Los proyectiles que lanza Samus no pueden atravesar paredes del escenario. Solo podrá disparar cuando no se encuentre en estado de esfera.

Samus cuenta con una barra de vida. El valor inicial de esta barra es de 30 puntos. El límite de esta barra es de 99 puntos. Si el valor de la barra de vida llega a cero o menos, Samus morirá.

En el escenario pueden encontrarse distintos tipos de enemigos. En esta versión generalizada sólo se utilizará un tipo de enemigo, los zoomers. Los zoomer son enemigos que se adhieren a paredes o plataformas, y se mueven alrededor de estas con una dirección y velocidad constante dadas.

Si dos zoomers se encuentran de frente, se atraviesan sin colisión entre ellos. Si un zoomer golpea a Samus, esta perderá ocho puntos de vida. Al ser golpeada por un zoomer, Samus tendrá un breve periodo de invencibilidad en el cual podrá atravesar a otros zoomers sin ser golpeada de nuevo.

Si un proyectil de Samus golpea a un zoomer dos veces, el zoomer será destruido. Al destruir un zoomer puede dejar un objeto que restaura cinco puntos de vida a Samus al tocarlo. En la figura 2.26 se observa un zoomer siendo destruido.

Figura 2.26: Samus destruyendo un zoomer en Metroid.

Para mayor información del videojuego y sus reglas, consultar [18].

2.1.5. Pokémon Rojo/Azul

Pokémon Rojo y Pokémon Azul son dos videojuegos de Rol por turnos desarrollados originalmente para la consola Game Boy en 1998. En general, estos videojuegos comparten las mismas reglas y mecánicas, sus diferencias son irrelevantes para este trabajo, por lo que pueden utilizarse sin distinción. En estos videojuegos, controlas al entrenador de pokémon Red, quien se encuentra en su viaje para vencer a todos los entrenadores de la liga pokémon [24].

El videojuego se divide en dos jugabilidades básicas, la exploración del mundo y las batallas. El jugador inicia con la jugabilidad de Exploración

del mundo, por lo que se explicará esta primero. En la figura 2.27 puede observarse una imagen del videojuego original en ejecución, en modo de exploración.

Figura 2.27: Pokémon Rojo/Azul en ejecución en modo exploración.

La exploración del mundo se realiza controlando a Red, de forma similar a la jugabilidad de The Legend of Zelda. Los escenarios se muestran en vista aérea. Red puede moverse hacia la izquierda, derecha, arriba o abajo, a una velocidad determinada.

En los escenarios se encuentran piedras, que delimitan el área por la que Red puede caminar. Red no puede atravesar las piedras, ni destruirlas o empujarlas.

En el escenario pueden encontrarse entrenadores rivales. Red no puede atravesar a los entrenadores rivales. Estos entrenadores tienen dos estados, sin enfrentar y enfrentados. Todos los entrenadores rivales inician en estado sin enfrentar. En estado sin enfrentar, los entrenadores rivales tienen una dirección hacia la que miran y una distancia de rango de visión.

Si Red camina enfrente de un entrenador rival, en la dirección a la que mira, a una distancia menor a la de su distancia de rango de visión y sin ningún obstáculo, Red se detendrá y el entrenador caminará en dirección a Red hasta encontrarse al lado de este, y comenzará una batalla. Si hay

una piedra u otro entrenador entre el entrenador rival y Red, el entrenador no se moverá ni iniciará la batalla.

Si Red se encuentra al lado de un entrenador rival en estado sin enfrentar, puede iniciar la batalla voluntariamente, incluso sin estar en su dirección hacia la que mira. En este caso, el entrenador rival no se moverá.

Cuando la batalla contra el entrenador rival termina, este se quedará en la posición a la que se movió y pasará al estado de enfrentado. Una vez en estado enfrentado, el entrenador ya no se moverá ni podrá entrar en batalla de nuevo. En la figura 2.28 se observa a un entrenador rival retando a Red.

Figura 2.28: Entrenador rival enfrentando a Red en Pokémon Rojo/Azul. Una vez derrotado, el entrenador rival no volverá a retar a Red.

Las batallas consisten en un enfrentamiento entre dos personajes, el jugador y un entrenador rival. Cada contrincante puede contar con un conjunto de entre uno y seis pokémon. Los pokémon son criaturas del videojuego. Los pokémon tienen un orden que puede ser alterado en cualquier momento fuera de la batalla. Al iniciar la batalla se elegirá como pokémon en batalla al que esté asignado como primero en el orden. En la figura 2.29 puede observarse una batalla.

Figura 2.29: Batalla en Pokémon Rojo/Azul.

Las batallas se componen de turnos. En cada turno cada contrincante elige un ataque de su pokémon actual y cuando ambos lo hayan seleccionado, se ejecutan los ataques y se pasa al siguiente turno. Cualquiera de los contrincantes puede elegir utilizar su turno para cambiar de pokémon por algún otro de su equipo. Para hacer esto es necesario que haya otro pokémon sin derrotar en el equipo.

No es posible gastar un turno en cambiar a un pokémon por el mismo que se tiene actualmente en batalla. La batalla termina cuando uno de los dos contrincantes termina con todos sus pokémon derrotados. Si el jugador termina con todos sus pokémon derrotados, pierde. En la figura 2.30 se observa la pantalla para cambiar pokémon, donde no se permite seleccionar al que está en batalla.

Figura 2.30: Pantalla de cambio de pokémon en Pokémon Rojo/Azul.

Cada pokémon cuenta con una especie, una barra de vida, un conjunto de estadísticas y un conjunto de entre uno y cuatro ataques. La especie del pokémon determina su tipo, el cual a su vez determina sus debilidades, fortalezas e inmunidades con respecto a los ataques. La especie también determina el conjunto de ataques que el pokémon puede aprender. El conjunto de ataques que puede aprender una especie de pokémon puede incluir ataques que no sean del mismo tipo que el pokémon.

La barra de vida es un valor que disminuye con cada ataque del pokémon rival, u ocasionalmente por un ataque del mismo pokémon que tenga el efecto de bajar energía al atacante. En esta versión generalizada sólo se utilizarán ataques que bajan por completo la barra de vida del propio pokémon, por lo que no es necesario explicar la forma en que se calcula el daño infligido por ataque, el cual depende de varios factores. Cuando la barra de vida de un pokémon llega a cero, se considera derrotado y no podrá usarse en batalla hasta curarlo.

Aunque en el videojuego original cada pokémon cuenta con varias estadísticas, en esta versión generalizada sólo será importante una, la velocidad. La velocidad de los pokémon indica cuál ataque se ejecutará primero en un turno. El ataque del pokémon con mayor velocidad se ejecutará primero. Si un ataque causa el final de la batalla antes de que se ejecute el ataque del otro pokémon, este otro ataque ya no se ejecutará.

Los ataques, al igual que las especies pokémon, tienen un tipo. Existe una tabla completa de tipos, debilidades, fortalezas e inmunidades entre tipos

de pokémon y ataques, pero en esta versión generalizada se utilizará sólo una inmunidad. Los pokémon de tipo fantasma son inmunes a los ataques de tipo normal. Por esto mismo si un pokémon fantasma recibe un ataque normal, no recibirá daño.

Cada ataque de un pokémon cuenta con un valor denominado PP. El PP determina el número de veces que este ataque puede ser utilizado por un pokémon. Cada uso resta uno al valor de PP. Cuando un ataque de un pokémon se queda sin PP, ya no puede ser utilizado.

Existen objetos que el jugador puede cargar para curar los puntos de vida o PP de los pokémon. Es importante mencionar su existencia, pues de incluirlos alterarían el estado del videojuego. En esta versión generalizada se supondrá que Red no tiene ningún objeto, o que estos objetos no existen.

Una vez explicadas las reglas generales, hace falta explicar un conjunto de pokémon que pueden ser construidos con las características del videojuego original. En el videojuego existen muchas especies de pokémon, pero en esta versión generalizada sólo es necesario explicar tres, Gastly, Electrode y Snorlax. Los Gastly son pokémon de tipo fantasma, mientras que los Electrode y Snorlax son pokémon de tipo normal.

De la misma manera, existen muchos ataques en el videojuego original, pero en esta versión generalizada sólo es necesario explicar uno, autodestrucción. El movimiento autodestrucción reduce los puntos de vida del pokémon atacante a cero y daña al oponente. El movimiento autodestrucción es del tipo normal. Todos los pokémon mencionados pueden aprender el movimiento autodestrucción.

Para mayor información del videojuego y sus reglas, consultar [24].

2.1.6. Mario Kart 64

Mario Kart 64 es un videojuego de carreras desarrollado originalmente para la consola Nintendo 64 en 1996. En este videojuego, el jugador controla a un personaje del universo de Super Mario Bros. en un vehículo y el objetivo de cada escenario es llegar a la meta después de dar un determinado número de vueltas [20]. En la figura 2.31 puede observarse una imagen del videojuego original en ejecución.

Figura 2.31: Mario Kart 64 en ejecución.

Este videojuego se desarrolla en un escenario 3D. El vehículo puede moverse hacia delante o en reversa con velocidades máximas dadas. La velocidad de reversa es menor a la velocidad de moverse hacia adelante. También se puede girar hacia la izquierda o la derecha, lo que provocará que al moverse, el vehículo se mueva en diagonal hacia la dirección en la que se gira.

Todos los escenarios son cíclicos, con el punto inicial y el final siendo el mismo. Cada que el personaje vuelva al punto final habiendo pasado por toda la pista, se contará como una vuelta.

En esta versión generalizada se utilizarán escenarios con las características de uno existente en el videojuego original, la pista arcoiris. En este escenario, se tiene un camino estrecho, con barda en algunas secciones que funcionan como paredes, y sin bardas en algunas otras secciones. Las bardas impiden al jugador avanzar por fuera de la pista. En las secciones sin barda, el jugador saldrá de la pista. Si el jugador sale de la pista, será devuelto a una posición anterior a donde salió de la misma.

En el escenario pueden encontrarse plataformas a distintos niveles y plataformas inclinadas. Esto permite que en general, la pista pueda tener partes

una encima de la otra a distintas elevaciones sin problemas. En la figura 2.32 se observan dos plataformas a distinta elevación.

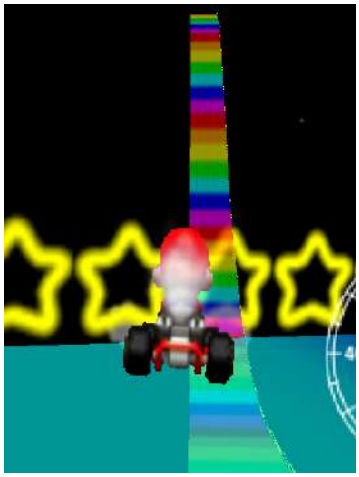


Figura 2.32: Plataformas a distinta elevación en Mario Kart 64.

Para esta versión generalizada se supondrá que dado un camino, es posible calcular en tiempo polinomial el tiempo óptimo necesario para recorrerlo. Si bien esta suposición no está comprobada para el videojuego original, en [4] se hace una conjetura de que discretizando el tiempo en intervalos de duración fija y describiendo a la pista como una sucesión de segmentos y vueltas, se puede llegar en tiempo polinomial a una aproximación lo suficientemente cercana al tiempo óptimo para el propósito de la demostración.

El jugador tiene un espacio para objetos. En el escenario pueden encontrarse cajas con objetos. Cada caja puede darte un objeto aleatorio. En esta versión generalizada se supondrá que una caja puede tener un objeto dado. Si el jugador no tiene ningún objeto en el espacio de objetos, puede tomar el objeto de la caja al tocarla.

Si el jugador tiene un objeto, puede utilizarlo en cualquier momento, este se consumirá del espacio de objetos al usarlo. Si el jugador tiene un objeto en su espacio de objetos, no podrá obtener otro objeto de una caja de objetos. En la figura 2.33 se observa una caja de objetos y su funcionamiento.

Figura 2.33: Cajas de objetos en Mario Kart 64.

Al utilizar un objeto, puede mantenerse detrás del vehículo sin lanzarlo. Si un jugador está manteniendo un objeto, puede tomar otro objeto de una caja de objetos. Aún así, al mantener un objeto no se podrá devolver al espacio de objetos, por lo que debe utilizarse antes del objeto que se encuentra en el espacio de objetos. Sólo se puede mantener un objeto a la vez.

De los objetos existentes en los videojuegos, se utilizarán dos para esta versión generalizada, los plátanos y los caparazones verdes.

Cuando un jugador utiliza un plátano puede lanzarla hacia enfrente o hacia atrás, a una distancia fija del vehículo. Si un plátano cae en el suelo al ser lanzada, se quedará en ese lugar. Si un plátano cae en un jugador, o toca una del suelo, recibirá un golpe y el plátano desaparecerá, lo que retrasa al jugador una pequeña cantidad de tiempo. En la figura 2.34 se observa un plátano y su funcionamiento.

Figura 2.34: Plátanos en Mario Kart 64.

Cuando un jugador utiliza un caparazón verde puede lanzarlo hacia enfrente o hacia atrás. Al lanzarla, el caparazón verde seguirá la dirección en la que fue lanzado. Si un caparazón choca con una pared, rebota en dirección contraria. Si un caparazón verde cae por el borde del escenario, caerá en diagonal siguiendo su dirección y la gravedad. Si un caparazón verde golpea un plátano, se destruirán ambos objetos.

Si un caparazón verde toca a un jugador recibirá un golpe y el caparazón desaparecerá, lo que retrasa al jugador una pequeña cantidad de tiempo. Si un caparazón verde no golpea a ningún objeto o personaje desaparecerá después de un tiempo dado. En la figura 2.35 se observa un plátano siendo destruido por un caparazón.

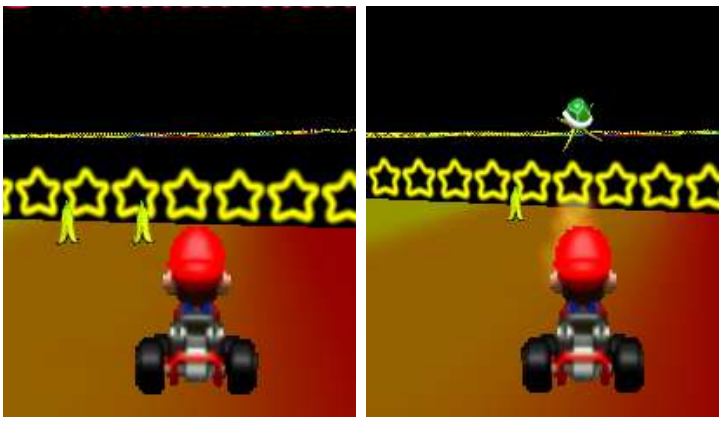


Figura 2.35: Caparazón destruyendo un plátano en Mario Kart 64.

En el videojuego original existen una variedad de modos de juego. En esta versión generalizada se utilizarán dos, el modo contrarreloj y el modo versus.

En el modo contrarreloj solo se puede jugar por un jugador a la vez. En el videojuego original, cuando se juega en este modo, no pueden encontrarse cajas de objetos en los escenarios. Para esta versión generalizada se supondrá que si. El objetivo de este modo de juego es dar una determinada cantidad de vueltas en el menor tiempo posible.

En el modo versus, pueden jugarse entre dos y cuatro jugadores. Todos los jugadores inician a moverse al mismo tiempo, y tienen visibilidad de la posición los demás jugadores con un minimapa. El objetivo de este modo

de juego es dar una determinada cantidad de vueltas antes que los otros contrincantes.

Para mayor información del videojuego y sus reglas, consultar [20].

2.1.7. Portal

Portal es un videojuego de rompecabezas, plataformas y disparos en 3D, desarrollado originalmente para Microsoft Windows, xbox 360 y Playstation 3 en 2011. En este videojuego, el personaje principal Chell debe recorrer una serie de rompecabezas en distintas habitaciones, utilizando su pistola de portales, para conseguir su libertad [19]. En la figura 2.36 puede observarse una imagen del videojuego original en ejecución.

Figura 2.36: Portal en ejecución.

Aunque los portales son la mecánica principal del videojuego original, las demostraciones presentadas en este trabajo no las utilizan. De hecho, en algunas demostraciones este elemento podría interferir con el resultado final. Por lo tanto, en esta versión generalizada se supondrá que este elemento no existe.

Chell puede moverse por el escenario en cualquier dirección en 360 grados

a la redonda, con una velocidad determinada. También puede saltar a una altura y distancia determinadas.

A diferencia de otros videojuegos, existen herramientas oficiales que te permiten crear escenarios para la secuela Portal 2, que comparte reglas con el videojuego original. Para una mayor validez, tanto las reglas presentadas como las demostraciones posteriores intentarán apegarse, en la medida de lo posible, a lo que el creador de escenarios permite agregar.

En el escenario pueden encontrarse plataformas a distinta elevación. Esto permite crear caídas largas. Una caída larga es aquella en la cual se le permite al jugador bajar de una plataforma de mayor elevación a una de menor elevación, pero no se permite subir de regreso, pues el salto del personaje no es lo suficientemente alto. En la figura 2.37 se observan una plataforma a menor elevación, creando una caída larga.

Figura 2.37: Caída larga en Portal.

En los escenarios pueden encontrarse puertas. Una puerta puede encontrarse abierta o cerrada, y puede atravesarse si y sólo si se encuentra abierta.

En el escenario pueden encontrarse escaleras retráctiles. Estas escaleras son plataformas que cuando se encuentran en estado apagado, se encuentran a nivel del suelo, pero en estado activado se levantan para permitirle a Chell subir en diagonal hacia plataformas más elevadas.

En los escenarios pueden encontrarse botones temporales. Los botones pueden ser activados o desactivados por Chell si se encuentra al lado de uno. Un botón puede realizar cambios de estado a un elemento del nivel, como una puerta, una plataforma o escaleras retráctiles. Una vez activado,

el elemento afectado volverá a su estado original después de una cantidad determinada de tiempo.

En la herramienta de creación de escenarios no se permite asignar el tiempo en el que el botón permanece activado. En esta versión generalizada se supondrá que si es posible. Visualmente, el botón se encuentra unido al elemento que tiene asignado por una línea de puntos pintada por el suelo y las paredes. En la figura 2.38 puede observarse un botón temporal, que controla unas escaleras retráctiles.

Figura 2.38: Botón temporal que controla unas escaleras retráctiles en Portal.

En el escenario pueden encontrarse bloques. Chell puede tomar bloques y posicionarlos enfrente de ella. Chell puede subir sobre estos bloques para alcanzar algunas plataformas a mayor altura. En la figura 2.39 puede observarse un bloque siendo tomado por Chell.

Figura 2.39: Bloque siendo transportado por Chell en Portal.

En los escenarios pueden encontrarse botones sensibles al peso. Estos botones se encuentran a nivel del suelo. Estos botones se activan cuando Chell o un bloque se encuentran sobre él.

Un botón sensible al peso puede estar ligado a una cantidad no acotada de elementos del escenario. De igual manera, un elemento del escenario puede tener asociado uno o varios botones sensibles al peso. Mientras estos botones se encuentran activados, mantendrán activados elementos del escenario a los que esté ligado.

De la misma forma que con los botones temporales, visualmente, el botón sensible al peso se encuentra unido a los elementos que tiene asignado por líneas de puntos pintadas por el suelo y las paredes. En la figura 2.40 puede observarse un botón a presión que controla una puerta.

Figura 2.40: Botón a presión en Portal. Puede ser presionado por Chell o por un bloque.

En el escenario pueden encontrarse enemigos llamados torretas. Las torretas no pueden moverse por sí mismos y tienen un campo de visión. Si Chell camina por el campo de visión de una torreta, este comenzará a dispararle a Chell. Si Chell recibe varios disparos de una torreta en un corto periodo de tiempo, morirá. En la figura 2.41 se observa una torreta atacando a Chell.

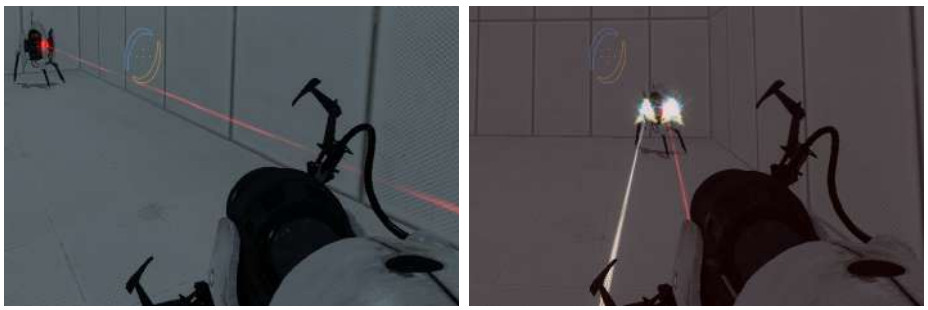


Figura 2.41: Torreta atacando a Chell en Portal.

Chell puede desactivar a una torreta si se encuentra cerca de él y fuera

de su campo de visión. Una vez que una torreta está desactivada, ya no volverá a causar daño a Chell. En la figura 2.42 se observa una torreta desactivada.

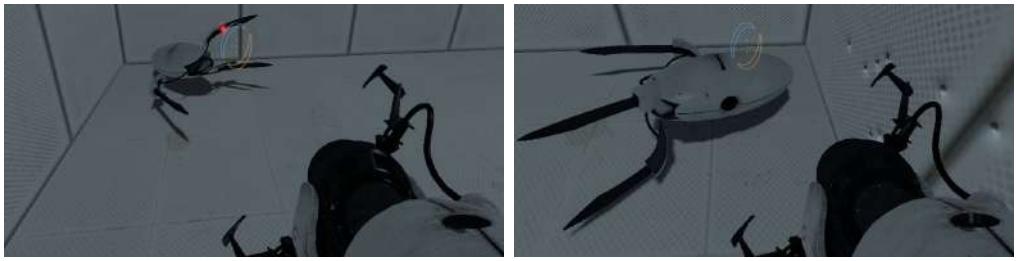


Figura 2.42: Torreta desactivada en Portal.

Para mayor información del videojuego y sus reglas, consultar[19].

2.1.8. Tetris

Tetris es un videojuego de rompecabezas desarrollado originalmente por Alexey Pajitnov para distintas plataformas en 1984. Actualmente se encuentra disponible para casi todas las consolas existentes en distintas versiones. En este videojuego, el jugador debe acomodar tetraminós (figuras conformadas por cuatro cuadros) en un tablero para destruirlos al apilar una fila completa[22]. En la figura 2.43 se observa el videojuego en una de sus versiones en ejecución. Las imágenes para ejemplificar corresponden a la versión de Game Boy.

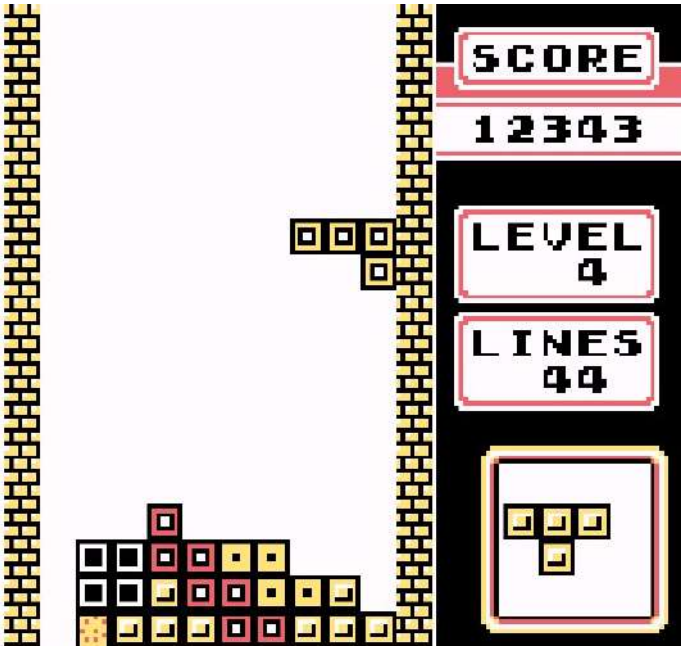


Figura 2.43: Tetris en ejecución.

En este videojuego, el escenario consiste en una cuadrícula rectangular, con algunos cuadros llenos y otros vacíos. Si bien en el videojuego original se tenían un tamaño único del escenario de 20×10 cuadros, en esta versión generalizada se supondrá que el escenario puede medir $m \times n$ para cualquier m,$n \in \mathbb{N}$. Cada fila se contará de abajo hacia arriba y cada columna de izquierda a derecha, comenzando ambos contadores con 0.

En una configuración legal del escenario, no puede haber filas con solamente cuadros llenos. De la misma forma no puede haber filas con solamente cuadros vacías que tengan filas con cuadros llenos por encima.

En cada partida se genera una secuencia aleatoria de tetraminós que se denominan como piezas. Cada pieza de la secuencia le es presentada al jugador en cada turno. En algunas versiones del videojuego, se le presenta al jugador una ventana de visibilidad de una pieza extra mostrándole la pieza que vendrá en el siguiente turno. En esta versión generalizada se supondrá que esta secuencia puede ser dada como entrada, y el jugador conoce desde un inicio la secuencia.

En cada turno, la siguiente pieza de la secuencia aparece en el centro en la fila superior del escenario. Una vez que la pieza aparece, bajará por el escenario a una velocidad dada. El jugador puede rotar la pieza hacia la

izquierda o a la derecha con respecto a su centro, así como moverla hacia la izquierda o la derecha. Todos los movimientos que realizan las piezas respetan la cuadrícula del escenario.

Las piezas no pueden atravesar los cuadros llenos del escenario. De la misma manera, las piezas no pueden realizar un giro que deje alguno de sus cuadros sobre un cuadro lleno del escenario. En la figura 2.44 se observan las piezas utilizadas en el videojuego con sus centros. La piezas serán nombradas, en el mismo orden que aparecen en la imagen, como C, Li, Ld, Si, Sd, l y T.

Figura 2.44: Piezas de Tetris con su centro.

Una pieza se detiene cuando cae en un cuadro lleno o cae en el suelo del escenario. Al detenerse, los cuadros que correspondían a la pieza se convierten en cuadros llenos. Si en alguna fila i todos los cuadros se encuentran llenos, la fila i desaparecerá, y todas las filas superiores bajarán. De esta forma, todas las filas j se convierten en la fila $j + 1$, con la última fila convirtiéndose en una fila vacía, para toda $i \leq j \leq m$.

Una vez que una pieza se detiene, el turno termina y empezará el siguiente apareciendo la siguiente pieza en pantalla. En la figura 2.45 se observa el proceso de eliminación de una fila.

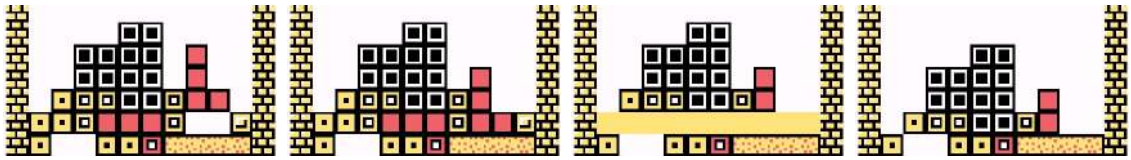


Figura 2.45: Eliminación de una fila en tetris.

El jugador perderá cuando los cuadros llenos del escenario impidan a nuevas empiezas aparecer en él. En general no se puede ganar en tetris, pero se pueden establecer metas para el mismo. Algunas de estas metas pueden ser conseguir una cantidad mínima de filas completadas, sobrevivir el mayor

número de turnos o dejar la pantalla con solamente cuadros vacíos, también llamado limpiar la pantalla.

Para mayor información del videojuego y sus reglas, consultar [22].

2.2. Técnicas generales para prueba de dureza computacional

En esta sección se presentarán algunas técnicas generales utilizadas para realizar reducciones a problemas conocidos que son *NP-Duros* y *PSPACE-Duros*. Estas técnicas se utilizarán en una gran cantidad de las demostraciones de este trabajo, pero también se presentarán demostraciones que no las utilizan. Dentro de estas técnicas, se utilizan estructuras específicas para la construcción de la reducción. Estas estructuras se detallarán en cada una de lás técnicas, así como sus componentes.

2.2.1. NP-Dureza

2.2.1.1. Reducción de 3-SAT

En [1] se define una estructura general para la reducción de *3-SAT* a varios videojuegos. El problema *3-SAT* fue demostrado como *NP-Completo* en [15]. El problema de decisión consiste en lo siguiente:

Entrada: Una fórmula booleana ϕ en 3-forma normal conjuntiva, con conjunto de variables $V = \{x_1, x_2, ..., x_n\}$ y cláusulas $C = \{c_1, c_2, ..., c_m\}$.

Problema de decisión: Decidir si existe una asignación de valores $\forall x \in V$ tal que satisface todas las cláusulas $c \in C$.

Al construir la estructura de esta técnica, se busca en general saber si dado un nivel de un videojuego con sus reglas, es posible llegar del punto inicial al punto final. La estructura de esta técnica se inspira fuertemente en las demostraciones de [7], en la cual se presenta demostraciones de *NP-Dureza* de los videojuegos PushPush y Push-1.

En la figura 2.46 se observa un diagrama que ejemplifica de manera general el funcionamiento de la estructura. Las flechas de un sólo sentido representan que no hay camino de vuelta, mientras que las dobles flechas representan que hay camino de ida y vuelta.

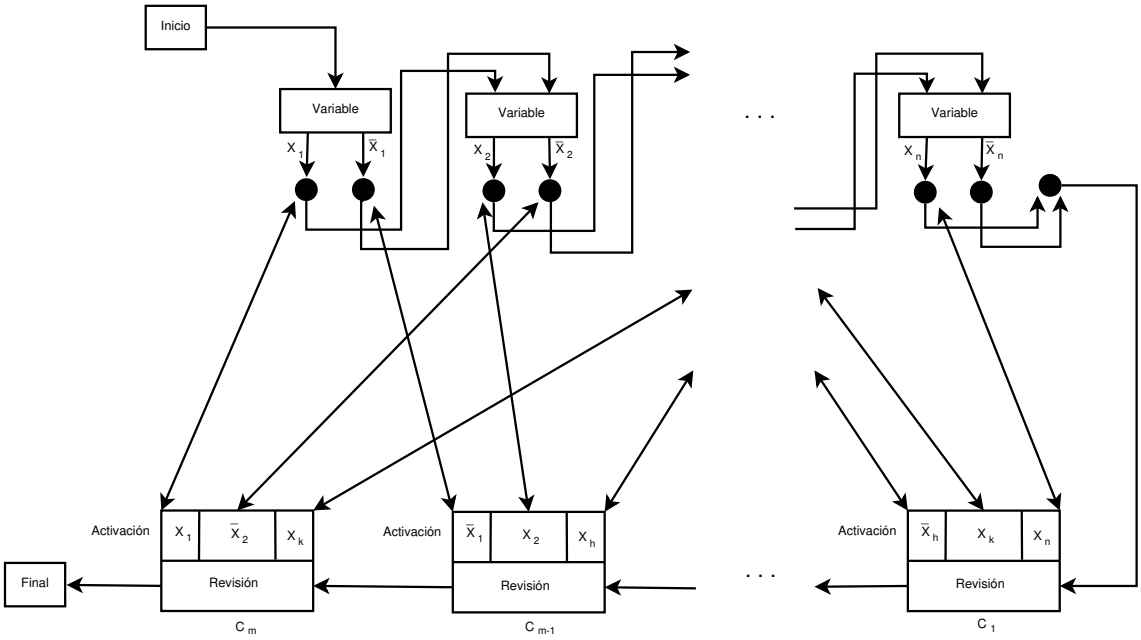


Figura 2.46: Estructura utilizada en la técnica de reducción de *3-SAT*.

Para implementar entonces esta estructura, basta con explicar como construir cinco tipos de componente: componente de inicio, componente de final, componente de variable, componente de cláusula y componente de cruce. A continuación se explicará el funcionamiento de cada uno.

- **Componente de inicio:** En algunos videojuegos se requiere que el jugador atraviese la estructura con el videojuego en cierto estado, por lo que este componente inicializa el videojuego en este estado.

  En algunos casos el estado del videojuego no influye al momento de atravesar la estructura, por lo que este componente no siempre es necesario, se puede suponer que al llegar al inicio de la estructura, el jugador llega con el estado necesario, pues la reducción sólo requiere que exista la posibilidad de que se llegue en ese estado.

- **Componente de final:** Este componente representa la meta a obtener, dependiendo del videojuego se puede requerir que se llegue hasta

este componente en cierto estado. Al igual que con el componente de inicio, el estado no siempre influye en que el jugador pueda o no terminar un nivel, por lo que este componente tampoco es siempre necesario.

- **Componente de variable:** Son secciones del nivel que se recorren en orden, corresponden a cada variable que aparece en la expresión booleana. En cada uno hay una división en la cual se obliga a elegir entre dos caminos, sin oportunidad de regresar para cambiar la elección.

  Esta elección representa a la asignación de valor a la variable, por lo que elegir el primer camino representa asignar valor positivo a la variable y elegir el segundo camino representa asignarle valor negativo.

- **Componente de cláusula:** Este componente se divide en dos secciones disjuntas, la sección de activación y la sección de revisión.

  La sección de activación se compone de tres entradas disconexas, conectadas cada una a cada camino de la literal de la expresión booleana original, en cada entrada se encuentra algún elemento que "abre" la sección de revisión.

  Las secciones de revisión de todos los componentes de cláusulas están conectados en secuencia, y en principio están bloqueadas. La única forma de pasar por ellas es previamente haber abierto el camino desde la parte de activación. De esta manera, el camino es transitable si y sólo si existe una asignación que satisface a todas las cláusulas.

- **Componente de cruce:** Como se puede notar en el ejemplo, no puede asegurarse que los caminos que conectan a los componentes de variable con los de cláusulas no tengan cruces entre ellos, por lo que se requiere un componente que permita transitar por cada uno de los caminos cruzados sin interferir entre ellos.

  Este componente en general sólo es necesario para videojuegos en 2D. En los videojuegos en 3D, basta con que los niveles permitan crear caminos cerrados a diferentes elevaciones para asegurar que no se cruzaran.

Hay dos observaciones importantes sobre el componente de cruce. Dada la construcción de la estructura, puede suponerse que sólo se va a utilizar en una dirección, por lo que basta con que cada camino sea unidireccional.

Por otro lado, como en general ambos caminos se van a transitar una sóla vez, y en todo cruce se sabe cuál de los dos caminos se atravesará primero, por lo que teniendo los caminos A y B, donde A es el camino que se transita primero, puede permitirse que haya filtración de A a B después de haber transitado B.

En total, se necesitan construir n componentes de variable, m componentes de cláusula, y a lo más un número polinomial de componentes de cruce. El número de componentes de cruce es porque se puede construir la estructura de tal forma que todos los caminos creados se crucen a lo más una vez entre ellos, y el número de caminos es $O(n+m)$. Los caminos que conectan a los componentes se pueden construir de tal forma que midan a lo más $O(n+m)$. Dado que cada componente es de tamaño constante, y se construye en tiempo constante, la reducción puede realizarse en tiempo polinomial.

Una vez dada la construcción de la estructura, hace falta demostrar que dada una fórmula booleana ϕ descrita como en la entrada del problema, existe una asignación de valores para toda $x \in V$ tal que satisface todas las cláusulas $c \in C$ si y solo si se puede terminar un nivel creado a partir de la fórmula ϕ con la descripción de la estructura.

Si existe esta asignación, llamemosla As, y dado que cada componente de variable contiene dos salidas que corresponden cada una a una asignación de la variable, se puede tomar la salida de la asignación As en cada variable. Al tomar esta salida, permite abrir todos los componentes de cláusula. Esto sucede porque cada salida de los componentes de variable tiene conexión con la sección de activación de los componentes de clásula que contienen su asignación correspondiente. Como se abrieron todos los componentes de cláusula, cuando el jugador llegue a la sección de revisión de los componentes, podrá pasar sin obstáculos. Por lo tanto, se puede terminar el nivel.

Por otro lado, para que el nivel se pueda terminar, es necesario que exista una forma de abrir todos los caminos de los componentes de cláusula. Dado que los componentes de cláusula solo se pueden abrir desde su sección de activación, que cada sección solo puede ser visitada desde una de las salidas de un componente de variable, que corresponde a una asignación

de la variable, y que las salidas de los componentes son excluyentes entre si, existe una sucesión de selecciones de salidas en los componentes de variable que permiten abrir todos los componentes de cláusula. Esta sucesión de selecciones de salidas corresponde a una asignación de variables que ϕ, por la naturaleza de la construcción.

Por lo tanto, la reducción es correcta.

Dado que la reducción es correcta y puede realizarse en tiempo polinomial, solo hace falta definir los componentes mencionados en tiempo y tamaño constante para poder realizar una reducción y demostrar la *NP-Dureza* de un videojuego.

2.2.1.2. Reducción de ciclo hamiltoniano en una gráfica en cuadrícula

En[8], se presenta una estructura general para la reducción desde el problema de decidir si existe un ciclo hamiltoniano en una gráfica en cuadrícula a un grupo de videojuegos con ciertas características. Este problema se encuentra demostrado como un problema *NP-Completo* en[14]. Este problema de decisión consiste en lo siguiente:

Entrada: Una gráfica $G = (V, A)$ en cuadrícula.

Problema de decisión: Decidir si existe un ciclo $H \subset G$ tal que para toda $v \in V$, v aparece exactamente una vez en H.

En[8], se presenta que un videojuego de plataformas con puertas controladas por interruptores temporales es *NP-Duro*. Aunque no se menciona en[8], la construcción requiere que el videojuego sea de plataformas 3D o de vista aérea para una reducción directa a la presentada. También se requiere que el tiempo que un interruptor permanece encendido pueda tener un valor dado específico.

Para la estructura de esta técnica, se construye un escenario con cuartos con interruptores en el centro posicionados en forma de cuadrícula con pasillos uniendolos. Cada cuarto representa a un nodo y cada pasillo representa una arista.

El tiempo que toma activar el interruptor y atravesar el cuarto, se denota

con D. Por otro lado, cada pasillo es lo suficientemente largo para tardar $a > n \cdot D$, donde n es el número de nodos en la gráfica. La desigualdad anterior asegura que no importe la diferencia de tiempo entre girar o seguir de frente al atravesar una habitación.

Al lado de la habitación inicial, se encuentra una serie de n puertas bloqueando la salida, estas puertas corresponden a los interruptores de cada cuarto. De esta manera, es posible alcanzar la salida si y sólo si todos los interruptores están activados al mismo tiempo. Esto requiere que se pase por todos los cuartos al menos una vez.

Todos los interruptores se desactivan después de un tiempo $((a+D)\cdot(n))+E$, donde E es el tiempo que toma recorrer desde el centro del cuarto inicial a la salida por el pasillo de puertas. Dado que siempre se puede asignar una a mucho mayor que E, puede asegurarse que para que haya tiempo suficiente para llegar a la salida, se visite cada cuarto una sola vez. En la figura 2.47 se observa la estructura utilizado en la técnica de reducción de ciclo hamiltoniano en una gráfica en cuadrícula.

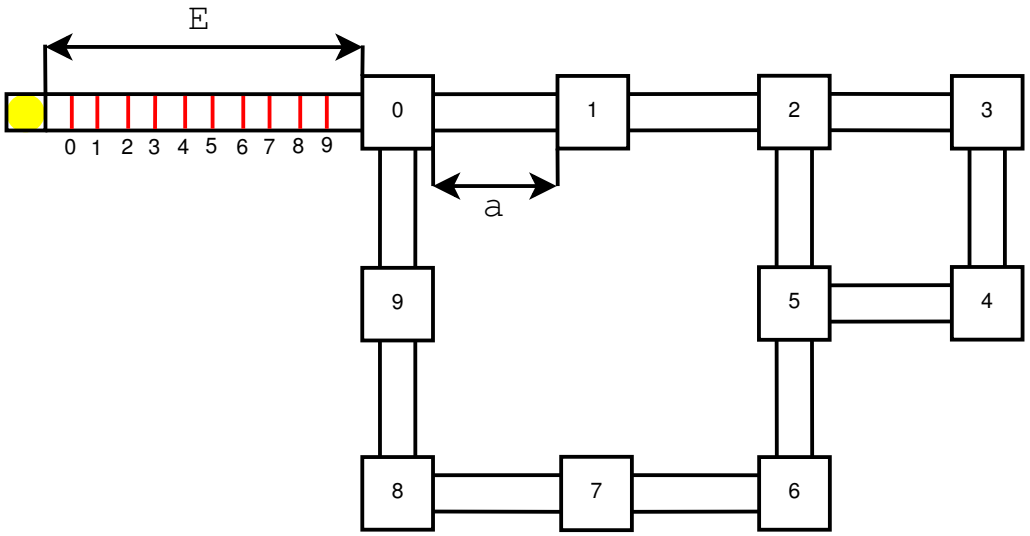


Figura 2.47: Estructura utilizada en la técnica de reducción de ciclo hamiltoniano en una gráfica en cuadrícula.

Cada cuarto utiliza un espacio constante y se construye en tiempo a lo más constante. El pasillo de puertas se construye en tiempo y toma espacio $O(|V|)$, pues depende del número de vértices para su construcción. De igual manera, cada pasillo que une a los cuartos se construye en tiempo y toma

espacio $O(|V|)$, y a lo más hay $O(|V|^2)$ pasillos. Por lo tanto, se puede construir la reducción en tiempo polinomial.

Una vez dada la construcción, se requiere que exista un camino tal que se visite exactamente una vez cada cuarto para que exista una forma de terminar el nivel. Por lo tanto, el nivel puede completarse si y solo si existe un ciclo hamiltoniano en G.

Dado que la reducción es correcta y puede realizarse en tiempo polinomial, solo hace falta definir los elementos específicos del videojuego con respecto a la estructura para poder realizar una reducción y demostrar la *NP-Dureza* de un videojuego.

2.2.2. PSPACE-Dureza

2.2.2.1. Reducción de TQBF

En[1] se utiliza una estructura general para la reducción de *TQBF* a algunos videojuegos. Este problema se encuentra demostrado como *PSPACE-Completo* en[2]. El problema consiste en lo siguiente:

Entrada: Una fórmula booleana cuantificada $\exists x_1 \forall y_1 ... \exists x_{n/2} \forall y_{n/2}$ $\phi(x_1, y_1, ..., x_{n/2}, y_{n/2})$ en forma normal conjuntiva.

Problema de decisión: Decidir si la fórmula es verdadera.

El problema es *PSPACE-Duro* incluso si se limita a las instancias en que los cuantificadores están intercalados iniciando con un cuantificador existencial, y sus cláusulas tienen exactamente tres literales. En esta técnica se utilizarán solamente instancias con estas características.

Al construir la estructura de esta técnica, al igual que en las técnicas para *NP-Dureza*, se busca en general saber si dado un nivel de un videojuego con sus reglas, es posible llegar del punto inicial al punto final. Esta estructura se presenta originalmente en[27], pero en este caso se utiliza con videojuegos con botones a presión. En[1] se extiende a una idea más general.

En la estructura de esta técnica se requieren cuatro componentes esencial-

mente, y una vez construidos estos, se pueden construir otros componentes más elaborados generales.

Los componentes que se requieren son el componente de inicio, el componente de final, componente de cruce y componente de puerta. Los primeros tres componentes son análogos a los mencionados en la estructura de la téncia para *NP-Dureza* de reducción de *3-SAT*. El cuarto componente, el componente de puerta, se explicará a continuación.

El componente de puerta consiste en tres caminos disjuntos. Estos caminos son llamados camino para transitar, camino para abrir y camino para cerrar. La parte que realmente representa a la puerta es el camino para transitar, por lo que de ahora en adelante se le llamará puerta a este camino. Esta puede ser atravesada si y sólo si la puerta se encuentra en estado abierto.

Los otros dos caminos, como su nombre lo indica, tienen la función de abrir y cerrar la puerta. Estos dos caminos tienen una pequeña diferencia, el camino para cerrar te obliga a cerrar la puerta, mientras que el camino para abrir te permite abrirlo, pero se puede decidir no hacerlo.

La decisión de que abrir la puerta sea opcional viene de que normalmente es natural para el jugador abrir un camino si no se tiene una consecuencia negativa al hacerlo, por lo que se puede suponer que decide siempre hacerlo.

En general, dejarlo como algo opcional para el jugador facilita la construcción del componente, además, posteriormente se observará que esto no influye en el funcionamiento del mismo. Dado que se establece como algo opcional el abrir la puerta, el camino para abrir puede simplificarse como un pasillo cerrado con una sola entrada que sirve también de salida. En la figura 2.48 se observa la estructura del componente de puerta.

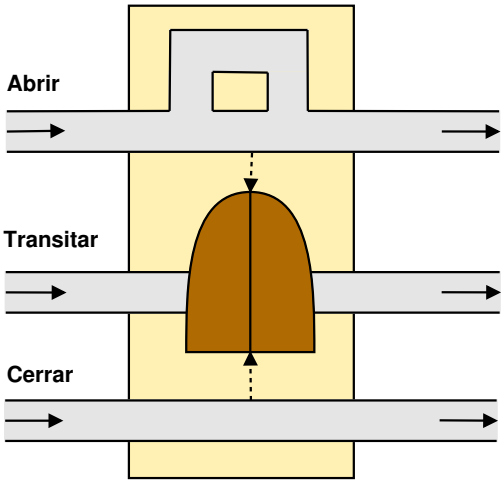


Figura 2.48: Componente de puerta para la estructura de reducción de *TQBF*.

Los componentes explicados anteriormente son suficientes para construir la estructura como tal, pero es necesario explicar su funcionamiento y la forma en que se acomodan los componentes. En la figura 2.49 se muestra un diagrama del funcionamiento general de la estructura.

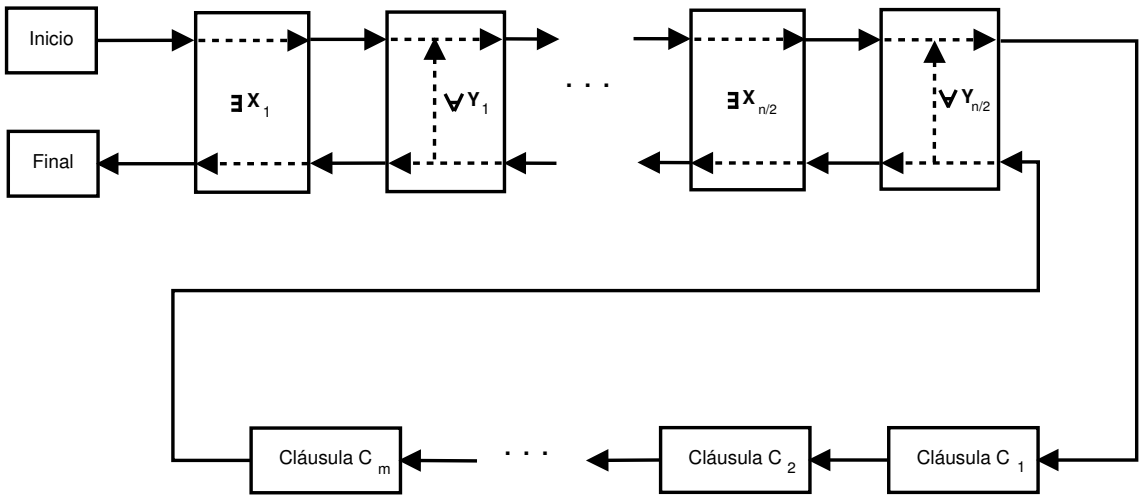


Figura 2.49: Estructura utilizada en la técnica de reducción de *TQBF*.

El funcionamiento general de la estructura es el siguiente: Dada una instancia de *TQBF*, con ϕ en 3-forma normal conjuntiva, se traduce primero en la sección de la estructura de componentes de cuantificadores (componente existencial y componente universal) seguido de la sección de los

componentes de cláusulas, conectadas por varios caminos mostrados en el diagrama de la figura 2.49.

Al atravesar hacia la derecha cada componente de cuantificador, se asigna un valor a la variable que corresponde al cuantificador. Al pasar por un componente existencial, se elige el valor de verdadero o falso. Por otro lado, al pasar por la parte superior de un componente universal, se le asigna el valor de verdadero a la variable correspondiente.

Una vez que todas las variables tengan un valor asignado, cada componente de cláusula se puede atravesar si y sólo si su cláusula correspondiente en ϕ se satisface con los valores asignados en este momento. Si se pudo atravesar esta sección, el jugador puede acceder a la parte inferior de los componentes de cuantificador.

En esta parte se redirige a la parte superior del último cuantificador existencial, pero en esta ocasión se le asigna el valor de falso y se vuelve a pasar por la sección de cláusulas. En esta ocasión de nuevo se puede atravesar si y sólo si ϕ se satisface con los valores actuales.

La próxima vez que se atraviese la parte inferior del cuantificador te dejará avanzar hacia la izquierda hasta llegar al cuantificador universal anterior. Aquí se le da el valor de falso a la variable correspondiente. Posteriormente se permite decidir un valor para la variable del último componente existencial y se asigna de nuevo el valor de verdadero a la variable del último cuantificador universal.

En este punto se tendrán que probar de nuevo todos los valores de la variable para avanzar. De esta forma continúa el proceso hasta dar todas las combinaciones de valores posibles a las variables de los cuantificadores universales y en cada caso, seleccionando valores indicados para las variables de los cuantificadores existenciales.

Este proceso simula la comprobación de los cuantificadores con las variables. De esta forma, la fórmula booleana es verdadera si y solo si el nivel se puede terminar.

Una vez explicado la estructura general, hace falta mostrar que se puede construir utilizando los componentes explicados al principio. El componente de inicio y componente de final son análogos a los de la estructura, por lo que basta mostrar que pueden construirse los componentes de cláusulas,

componentes existenciales y componentes universales utilizando solamente componentes de puertas y componentes de cruces.

A continuación se explicará la construcción de cada uno utilizando los componentes de puertas. Los componentes de cruce se utilizarán para conectar las estructuras sin que la planaridad afecte a la construcción.

- **Componente de cláusula:** El componente de cláusula es sencillo, se compone de un pasillo conectado a tres puertas distintas, correspondientes a las literales de las cláusulas, y conectadas del otro lado a otro pasillo para avanzar.

  De esta forma, este componente se puede atravesar si y sólo si al menos una de las puertas de las literales se encuentra abierta. En la figura 2.50 se observa la estructura del componente de cláusula, los recuadros grises corresponden a puertas de las literales de la cláusula.

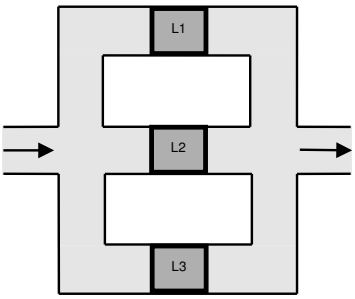


Figura 2.50: Componente de cláusula para la estructura de reducción de *TQBF*.

- **Componente existencial:** Siendo x la variable correspondiente al cuantificador existencial representado por el componente existencial. Se llamará x_i a la i-ésima aparición de la variable x en las distintas cláusulas de ϕ. Así mismo con $\overline{x}_i$. Al atravesar la parte superior del componente, se debe elegir entre dos caminos, que corresponden a asignar un valor a la variable correspondiente.

  Cada camino tiene al final de su recorrido una puerta que se cierra al seleccionar el camino contrario, de esta forma se evita transitar por ambos caminos a la vez. Si se selecciona, por ejemplo, el camino positivo, este inicia con una ruta al camino de cierre de la puerta negativa de este componente.

  Posteriormente se conecta a los caminos para abrir de $x_1, x_2, ..., x_m$.

Esto se conecta a los caminos para cerrar de $\overline{x}_1, \overline{x}_2, ..., \overline{x}_n$. Finalmente, se conecta al camino para abrir la puerta positiva de este componente, para poder avanzar.

De esta manera, al terminar el recorrido, todas las puertas correspondientes a apariciones positivas de la variable en las cláusulas están abiertas y todas las negativas cerradas. El recorrido por el camino negativo es análogo.

La parte inferior del componente es simplemente un pasillo que te permite transitar de derecha a izquierda. Este componente puede verse representado en la figura 2.51, la expresión $+x_i$ representa al camino para abrir la puerta correspondiente a x_i, mientras que $-x_i$ representa el camino para cerrar la puerta correspondiente a x_i, los recuadros grises corresponden a puertas.

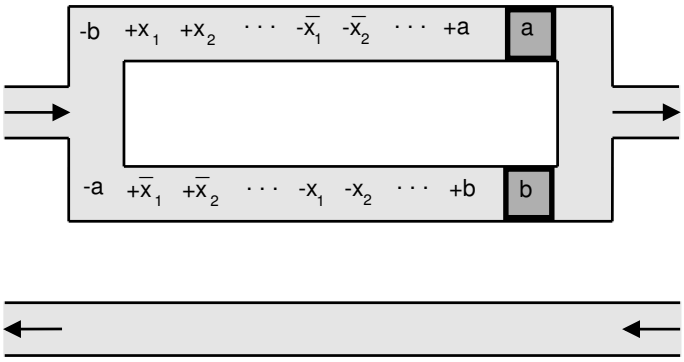


Figura 2.51: Componente existencial para la estructura de reducción de *TQBF*.

- **Componente universal:** El componente universal es algo más complicado, por lo que para explicarlo, se utilizará el esquema mostrado en la figura 2.52. Se utilizará la misma terminología y simbología que en el componente existencial.

  Primero se debe atravesar la parte superior del componente, con lo cual se tiene una ruta hacia el camino que cierra la puerta inferior d. Posteriormente se conecta a los caminos para abrir de $x_1, x_2, ..., x_m$. Esto se conecta a los caminos para cerrar de $\overline{x}_1, \overline{x}_2, ..., \overline{x}_n$.

  Finalmente se conecta al camino para abrir a, que permite avanzar. Una vez atravesada esta ruta, no es posible acceder a ninguna ruta,

pues al intentar pasar por el camino c se tiene que pasar por el camino para cerrarlo.

La segunda vez que se transita por el componente, se llega por la entrada inferior derecha. En este momento, no es posible transitar por la puerta d, pues se cerró este camino al pasar la primera vez por el componente, esto obliga a tomar el camino con la puerta b para avanzar.

Al atravesar la puerta b, se hace lo contrario que la primera vez que se transita por el componente. Primero se tiene una ruta hacia el camino que cierra la puerta por la que se entró.

Posteriormente se conecta a los caminos para abrir de $\overline{x}_1, \overline{x}_2, ..., \overline{x}_n$. Esto se conecta a los caminos para cerrar de $x_1, x_2, ..., x_m$. Posteriormente se abre la puerta inferior d y finalmente abre la puerta c, que permite avanzar. Al atravesar la puerta c se cierran las puertas c y a, evitando que se intente volver por alguna de esta puertas.

La siguiente vez que se transita por el componente, se llega de nuevo por la entrada inferior derecha, pero esta vez la puerta d está abierta, por lo que se puede avanzar por este camino. Cuando se continúa el proceso y se requiere pasar por el componente desde la entrada superior izquierda, se volverá a reiniciar el proceso.

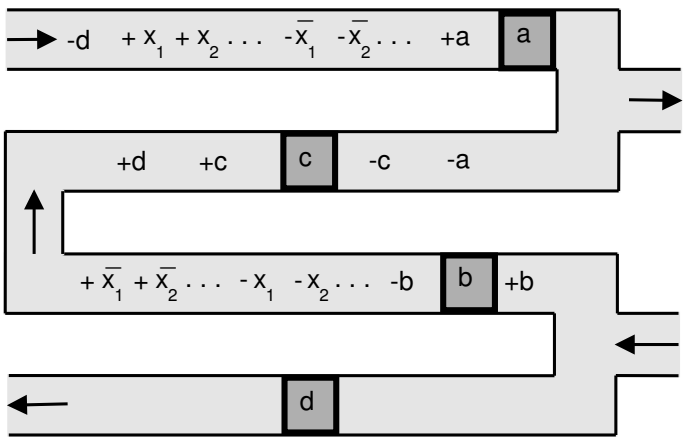


Figura 2.52: Componente universal para la estructura de reducción de *TQBF*.

Por la construcción de la estructura, cada puerta es accedida por una ruta a la izquierda y una hacia la derecha por cada uno de sus tres caminos

disconexos. Los caminos para abrir y para cerrar sólo tienen conexión con el componente de cuantificador correspondiente y el camino para transitar tiene conexión sólo con su cláusula.

Por otro lado, no es necesario que en general, las entradas de los caminos del componente de puerta sean siempre de izquierda a derecha. De hecho, en el caso del camino para abrir, nada impide que la entrada y la salida sean por el mismo camino. Esto significa que se puede simplificar como un simple callejón sin salida, si la reducción específica lo requiere.

Finalmente, cabe remarcar que el camino para abrir sólo abre nuevas áreas, sin cerrar ninguna. Por esto mismo es seguro suponer que el jugador siempre elegirá abrir la puerta desde el camino para abrir. En caso de que el jugador eligiera no abrirlo, y aún así pudiera llegar al final, forzosamente podría hacerlo aunque lo abriera.

Por otro lado, si el camino para llegar al final no existe cuando el jugador decide abrir las puertas, no hay ninguna forma posible de que el camino exista al decidir no abrir alguna puerta.

En total, se necesitan construir n componentes de variable entre las universales y las existenciales, m componentes de cláusula, y a lo más un número polinomial de componentes de cruce.El número de componentes de cruce es porque se puede construir la estructura de tal forma que todos los caminos creados se crucen a lo más una vez entre ellos, y el número de caminos es $O(n * m)$, m por cada variable y uno por cada cláusula. Los caminos que conectan a los componentes se pueden construir de tal forma que midan a lo más $O(n * m)$. Los componentes universales y los existenciales se construyen entiempo y tamaño $O(m)$, mientras que los otros componentes se construyen en tiempo y espacio constante. Por lo tanto, la reducción se puede realizar en tiempo polinomial.

Dado que la reducción es correcta y puede realizarse en tiempo polinomial, solo hace falta definir los componentes de puerta, cruce, inicio y final en tiempo y tamaño constante para poder realizar una reducción y demostrar la *PSPACE-Dureza* de un videojuego.

2.2.2.2. Reducción de lógica de restricción no determinista

En[8], se presenta una estructura general para la reducción desde el problema de lógica de restricción no determinista a un grupo de videojuegos con ciertas características. Este problema se encuentra demostrado como un problema *PSPACE-Completo* en[12]. Para entender el problema de decisión que se utilizará, primero se definirá lo que es una gráfica de restricción AND/OR.

Como se menciona en[8], una gráfica de restricción se define como la tupla $R = (G, w, c, p)$. En donde $G = (V, E)$ es una gráfica no dirigida.

El elemento w es una asignación de enteros no negativos a las aristas de G, $w : E \to \mathbb{Z}^+$, conocidos como pesos.

El elemento c es una asignación de enteros a los vértices de G, $c : V \to \mathbb{Z}$, conocidos como restricciones.

Cada arista tiene una orientación $p : E \to \{+1, -1\}$. La orientación p induce a una gráfica dirigida $D_{G_p} = (V, A)$.

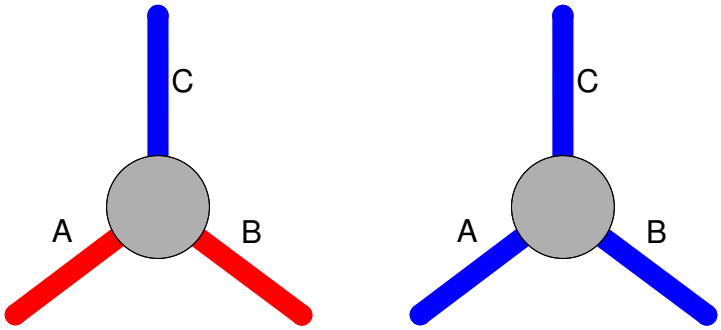
Sea $v \in V$ un vértice de G, la in-vecindad de v se refiere a los vértices adyacentes a v con una arista que apunta a v. La in-vecindad de v se define de la siguiente forma: $N^-(v, p) = \{w : (v, w) \in A\}$.

Como se menciona en[8], se dice que una gráfica de restricción es válida si, para toda $y \in V$, $\sum_{x \in N^-(y,p)} w((x, y)) \geq c(x)$. Se puede cambiar el estado de la gráfica seleccionando una arista y multiplicando su orientación por -1, de tal forma que la gráfica de restricción resultante sea válida. Se dice que la arista se ha girado.

Como se menciona en[8], se dice que un vértice v de una gráfica es un vértice AND si tiene exactamente tres aristas incidentes A B C, con restricción $c(v) = 2$, y con pesos $w(A) = w(B) = 1$ y $w(C) = 2$. Se le llama vértice AND porque para que C pueda apuntar hacia afuera de v, es necesario que ambos A y B apunten a v. En la figura 2.53a se observa un vértice AND, con las aristas de peso 1 en rojo y las aristas de peso 2 en azul.

De la misma manera, y como se menciona en[8], se dice que un vértice v de una gráfica es un vértice OR si tiene exactamente tres aristas incidentes A

B C, con restricción $c(v) = 2$, y con pesos $w(A) = w(B) = w(C) = 2$. Se le llama vértice OR porque para que C pueda apuntar hacia afuera de v, es necesario que al menos A o B apunten a v. En la figura 2.53b se observa un vértice OR, con las aristas de peso 2 en azul.



(a) AND **(b)** OR

Figura 2.53: Vértice AND y vértice OR de una gráfica de restricción.

Finalmente, como se explica en[8], una gráfica de restricción es AND/OR si todo vértice de ella es un vértice AND o OR. Es importante notar que en este tipo de gráficas, la restricción de todos los vértices es 2, y el peso de todas las aristas es 1 o 2.

Una vez explicado el concepto de gráfica de restricción AND/OR, se procederá a definir el problema de lógica de restricción no determinista. Esta demostración es tomada de[8].

Entrada: Una gráfica de restricción AND/OR $R = ((V, E), w, c, p)$, y una arista objetivo $(i, j) \in E$.

Problema de decisión: Decidir si existe una gráfica de restricción $R' = ((V, E), w, c, p')$ tal que $p'((i, j)) = -p((i, j))$, que puede ser obtenida a partir de R con una secuencia de giros de aristas.

La estructura utilizada en la técnica que se presenta a continuación, presentada en[8], realiza una reducción de un videojuego con puertas que puedan ser controladas por un interruptor, y con interruptores que puedan controlar al menos seis puertas al problema de lógica de restricción no determinista.

Cada arista se representa con un interruptor, en el cual el estado del mismo

representa la orientación de la arista. Cada interruptor se encuentra hasta el fondo de un componente al que se le llamará componente de revisión de consistencia. Cada componente se conecta a un pasillo central, que tiene acceso a todos los componentes de revisión de consistencia.

El componente de revisión de consistencia consiste en dos secciones, que corresponden a los vértices adyacentes a la arista. Cada sección contiene pasillos con puertas correspondientes a las aristas adyacentes al vértice. Si la arista apunta al vértice, estará abierta, de lo contrario estará cerrada.

El interruptor cambiará el estado de todas las puertas asociadas a él. Esto representa el giro de la arista, pues ahora ya no apuntará al vértice que antes sí, y apuntará al que antes no.

Para los vertices AND, se tiene una sección separada por dos pasillos. A las aristas de peso 1, les corresponde uno de los pasillos, el cual tendrá dos puertas, que son abiertas por el interruptor de cada una de las dos aristas. Al otro pasillo le corresponde la arista de peso 2, que contiene una puerta que se abre con el interruptor de la arista.

De esta forma, para atravesar esta sección con la puerta de la arista de peso 2 cerrada, si y sólo si las otras dos puertas de las aristas están abiertas. Esto indica que esta sección puede atravesarse si y sólo si la gráfica está en un estado válido para el vértice correspondiente a esta sección.

Para los vértices OR, se tiene una sección separada por cada pasillo. Cada arista adyacente al vértice tiene asignado un pasillo, con una puerta en el mismo que se abre con el interruptor de la arista correspondiente.

De esta forma, se podrá atravesar esta sección con una de las puertas cerrada si y sólo si al menos una de las otras dos está abierta. Esto indica que esta sección puede atravesarse si y sólo si la gráfica está en un estado válido para el vértice correspondiente a esta sección.

Es importante notar dos cosas. Cada giro de arista afecta a la validez de la gráfica solamente en sus vértices adyacentes. Además, como las secciones del componente de revisión de consistencia representan en sus pasillos a aristas adyacentes al vértice, y las secciones representan a los vértices adyacentes a la arista del componente, cada sección contiene una puerta que se controla con el interruptor.

Por estos dos hechos, se podrá volver al pasillo central de la estructura al presionar un interruptor si y sólo si, al girar la arista correspondiente en la gráfica de restricción, queda en un estado válido.

En la figura 2.54 se observa un ejemplo de un componente de revisión de consistencia de una arista específica. Las aristas rojas tienen peso 1, mientras que las azules tienen peso 2. La I representa al interruptor, las líneas negras representan puertas cerradas mientras que las azules representan puertas abiertas.

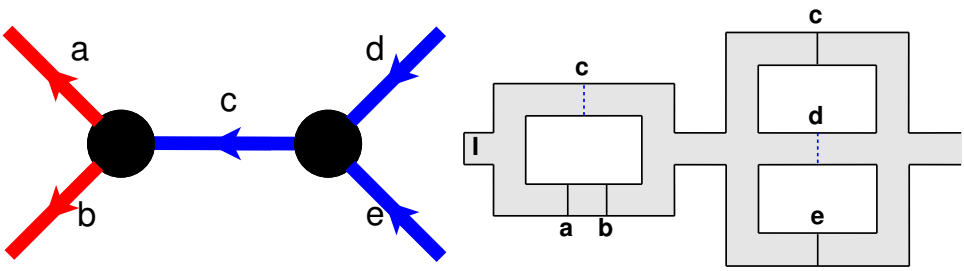


Figura 2.54: Componente de revisión de consistencia para la arista c de la estructura de reducción de lógica de restricción no determinista para *PSPACE-Dureza*.

Cada interruptor controla a exactamente seis puertas, siguiendo la construcción explicada. Esto es porque en su componente de revisión de consistencia aparecen dos puertas correspondientes a la arista, una por cada vértice adyacente, y en cada arista adyacente, las cuales son cuatro.

El interruptor de la arista objetivo es distinto a los demás. Este interruptor controla las puertas de su componente de revisión de consistencia y una puerta adicional en el pasillo central. Esta puerta conecta con la salida del nivel o el punto objetivo.

Este interruptor no necesita controlar las puertas de los componentes de revisión fuera del suyo, porque se presionará si y sólo si puede llegarse a girar la arista objetivo con movimientos válidos, dada la construcción explicada anteriormente. Una vez que sea presionado, la respuesta al problema ya estará dada, por lo que entrar a un componente de revisión de consistencia o presionar un interruptor después de esto, no tendrá efecto en la reducción.

De esta manera, dado que la construcción se traduce directamente de la

instancia original, se podrá llegar a la salida si y sólo si, es posible girar la arista objetivo.

Por otro lado, se utiliza una construcción de tamaño constrante por cada vértice de la gráfica de restricción, por lo que se puede construir la instancia del nivel a partir de una instancia del problema original en tiempo polinomial.

Dado que la reducción es correcta y puede realizarse en tiempo polinomial, solo hace falta definir los elementos específicos del videojuego con respecto a la estructura para poder realizar una reducción y demostrar la *PSPACE-Dureza* de un videojuego.

2.2.3. Pertenencia a PSPACE

Adicionalmente a la prueba de *PSPACE-Dureza*, en[1] se menciona que la mayoría de problemas de decisión en videojuegos se encuentran en *PSPACE*. Esto se debe a que la configuración de todos los elementos del escenario en la mayoría de los videojuegos se obtienen de una función determinista de los movimientos del jugador. Dado esto, toda configuración de un nivel puede ser obtenida realizando movimientos de manera no-determinista, manteniendo siempre el estado del videojuego.

Por otro lado, existen videojuegos que cuentan con enemigos con movimientos pseudo aleatorios. Si la semilla aleatoria de estos enemigos se puede codificar en un número de bits polinomial, el cual es el caso en las implementaciones razonables, su estado en cualquier momento puede obtenerse con una cantidad polinomial de iteración a partir del estado inicial y los movimientos del jugador.

De lo anterior, y del hecho de que $NPSPACE = PSPACE$ mencionado anteriormente, se sigue que si un videojuego cumple estas características, el videojuego está en *PSPACE*.

2.3. Resultados de dureza conocidos

Esta sección se dividirá en dos partes, primero se explicarán las demostraciones sobre *NP-Dureza* y posteriormente las de *PSPACE-Dureza* y *Completitud*. Cada una de estas secciones se dividirá a su vez en los videojuegos que utilizan una técnica descrita anteriormente y los que no. Para algunos videojuegos puede existir más de una demostración. En este caso se indicará qué técnica se utiliza o a partir de qué problema se hace la reducción en cada demostración.

El problema de decisión a resolver depende del videojuego a estudiar. En cada caso se indicará cuál es el problema de decisión a resolver.

Aunque en la mayoría de los casos no se menciona explícitamente, los escenarios creados tienen un tamaño polinomial. Esto es porque las estructuras que los componen tienen un tamaño constante o polinomial, y estas estructuras se repiten un número constante o polinomial de veces.

2.3.1. NP-Dureza con técnica

2.3.1.1. Super Mario Bros. (3-SAT)

En[1] se muestra la siguiente demostración de Super Mario Bros. generalizado, donde se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el problema de decisión a resolver es decidir si es posible terminar un nivel dado de Super Mario Bros. generalizado siguiendo sus reglas.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de inicio, componente de final, componente de variable, componente de cláusula y componente de cruce. Todos estos componentes se presentan en[1].

Componente de inicio:

Para el recorrido de esta construcción, se requiere Mario se encuentre en estado grande. Esto es por dos cosas, para evitar que pueda pasar a través de ciertos pasillos al medir 2 cuadro, y para permitirle romper ciertos bloques

de ladrillos. Para que Mario pueda obtener el estado grande, el componente de inicio contendrá un bloque de objeto con un super champiñón. El componente de inicio se observa en la figura 2.55.

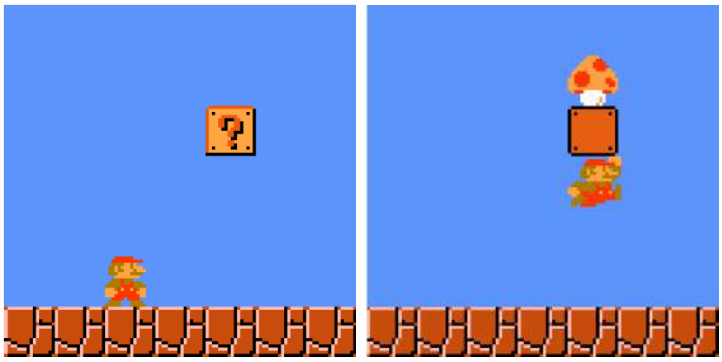


Figura 2.55: Componente de inicio de Super Mario Bros.para la estructura de *3-SAT*.

Componente de final:

El componente de final obliga al jugador a tomar el super champiñón del componente de inicio. Para esto, Mario llega a una zona cubierta de bloques simples con la excepción de un bloque de ladrillos. La única forma de avanzar es destruyendo el bloque de ladrillos. Para destruir el bloque de ladrillos Mario debe llegar en estado grande. El componente de final puede observarse en la figura 2.56.

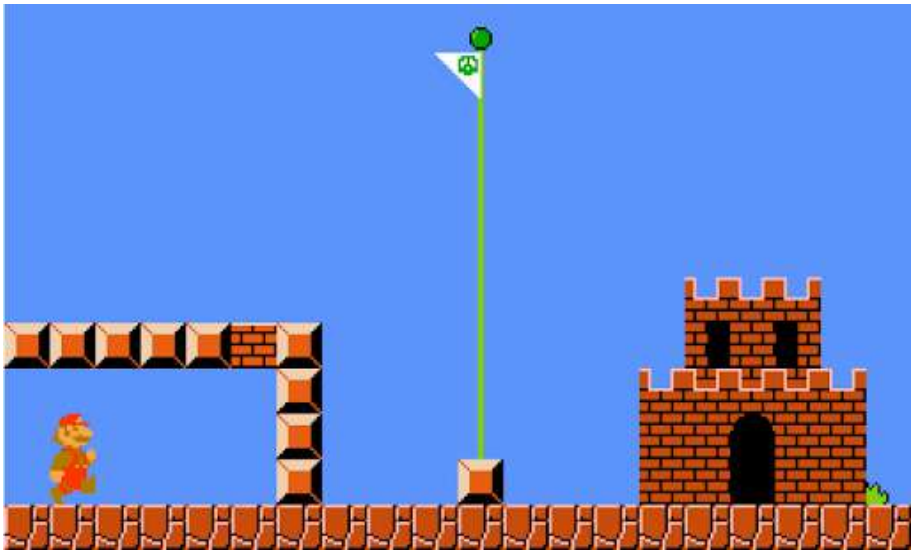


Figura 2.56: Componente de final de Super Mario Bros. para la estructura de *3-SAT*.

Componente de variable:

El componente de variable se observa en la figura 2.57. Para este componente se tienen dos entradas y dos salidas. Las entradas se encuentran en la parte superior y las salidas en la parte inferior. Una salida corresponde a la asignación positiva de la variable y la otra corresponde a la asignación negativa de la variable, siendo la de la izquierda la positiva y la derecha la negativa.

Cada una de estas salidas tienen conexión con una de las entradas de la sección de activación del componente de cláusula. Posteriormente la salida positiva se conecta a la entrada izquierda del componente de la siguiente variable, y la salida negativa se conecta con la entrada de la derecha.

Cuando se selecciona una de las salidas, se cae de una altura tal que no es posible regresar al componente de variable, por lo que no se pueden seleccionar ambas asignaciones.

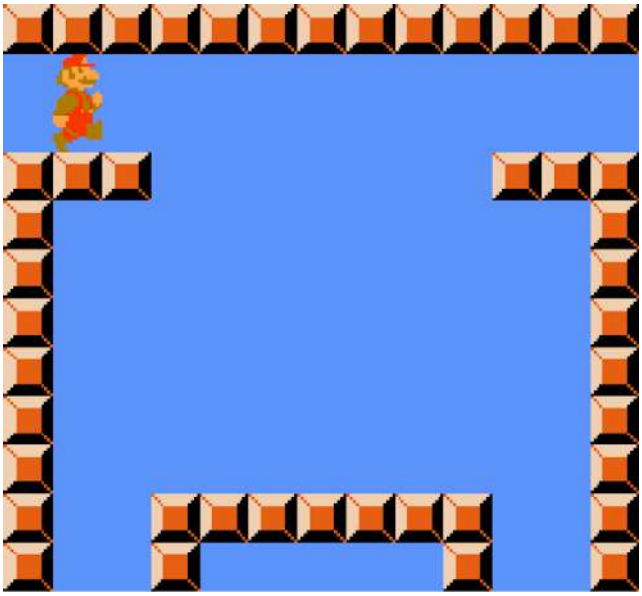


Figura 2.57: Componente de variable de Super Mario Bros. para la estructura de *3-SAT*.

Componente de cláusula:

El componente de cláusula se observa en la figura 2.58. La sección de activación de este componente se encuentra en la parte inferior. Como se puede observar en la figura 2.58, se tienen tres entradas con un bloque de objeto en cada una. Cada bloque de objeto contiene una estrella, que quedará rebotando en la parte superior.

La parte superior a los bloques está rodeada por dos paredes de bloques simples que impiden escapar a las estrellas. Mario no puede pasar de la sección de activación a la de revisión, ni puede pasar de una entrada a otra de la sección de activación.

La sección de revisión inicia con la sección donde se quedan las estrellas rebotando. Posteriormente se encuentra una sección muy larga repleta de barras de fuego, de las cuales es imposible pasar corriendo sin que toquen a Mario. De esta manera, la única forma de que Mario puede pasar a través de las barras de fuego, es con la invencibilidad de la estrella, la cual debe ser activada desde una entrada de la sección de activación.

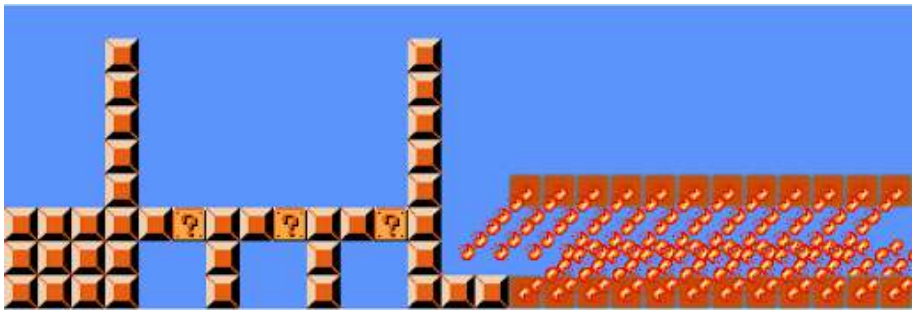


Figura 2.58: Componente de cláusula de Super Mario Bros. para la estructura de *3-SAT*.

Componente de cruce:

El componente de cruce se observa en la figura 2.59. Para este componente, Mario debe recorrer primero el camino de izquierda a derecha y posteriormente el de abajo hacia arriba. Como se mencionó en la descripción de la estructura, siempre es posible colocar el componente de tal forma que se recorra en este orden.

Para pasar de izquierda a derecha, Mario necesita estar en estado pequeño para poder pasar por el pasillo de bloques simples. Dado lo estrecho que son los pasillos creados por los bloques simples, Mario no podrá correr y agacharse para pasar por ellos en estado grande. Para esto, Mario puede recibir daño del goomba.

Como Mario se encuentra en estado pequeño mientras atraviesa ese pasillo, no puede destruir los bloques de ladrillos y no puede pasar al camino de abajo hacia arriba. Una vez que pasó por estos pasillos, encontrará un bloque de objetos que contiene un super champiñón. Mario debe tomar el super champiñón para destruir el último bloque de ladrillos y poder seguir avanzando.

Por otro lado, para pasar de abajo hacia arriba, Mario necesita estar en estado grande para romper los bloques de ladrillos que bloquean su camino. Como está en estado grande, Mario no puede pasar a la izquierda o derecha para entrar al camino que va de izquierda a derecha.

Si bien esto crea una forma de pasar de la entrada izquierda al camino de abajo hacia arriba, por la construcción no se pasará por la entrada de la izquierda después de pasar por el camino de abajo hacia arriba.

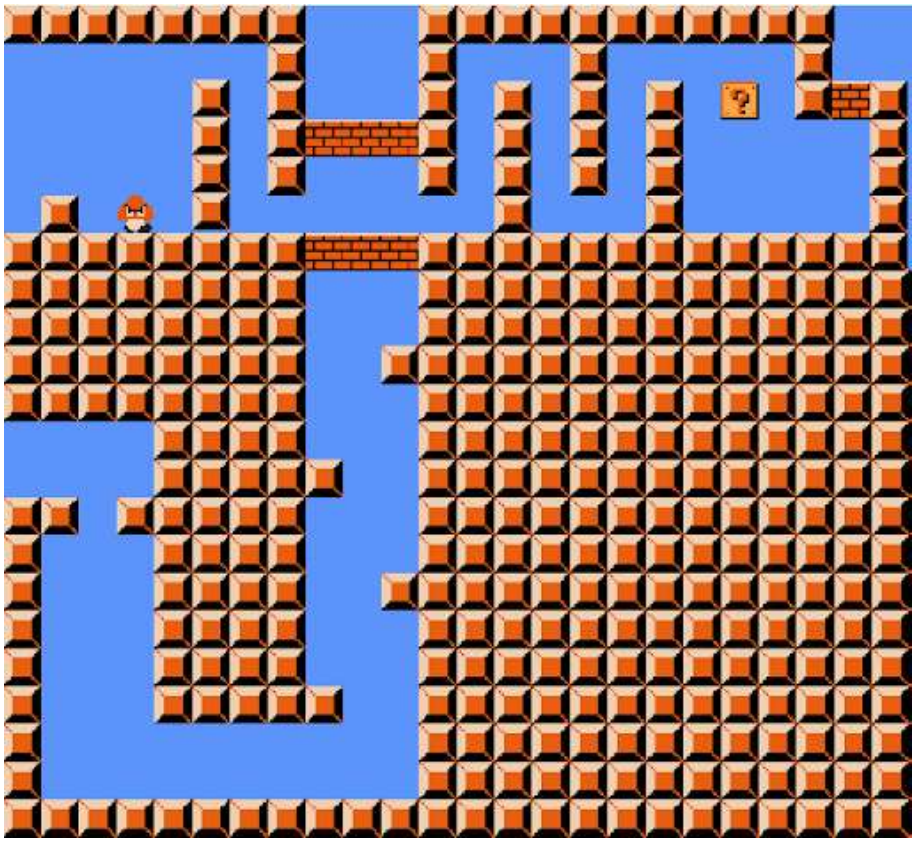


Figura 2.59: Componente de cruce de Super Mario Bros. para la estructura de *3-SAT*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si es posible terminar un nivel dado de Super Mario Bros. generalizado siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en[1] se presentan demostraciones con esta técnica para algunas secuelas de Super Mario Bros. Sin embargo, estas demostraciones dependen de algunas reglas introducidas en las mismas secuelas.

2.3.1.2. Donkey Kong Country (3-SAT)

En[1] se muestra la siguiente demostración de Donkey Kong Country generalizado, donde se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el problema de decisión a resolver es decidir si es posible terminar un nivel dado de Donkey Kong Country generalizado siguiendo sus reglas.

Si bien la demostración de *NP-Dureza* de Donkey Kong Country presentada en[1] menciona que el componente de inicio sólo corresponde al inicio de nivel, en la demostración de *PSPACE-Completitud* del mismo videojuego se menciona un componente de inicio que también es útil para esta demostración.

El componente de final sólo contiene la meta del nivel. El componente de variable es análogo al presentado para Super Mario Bros. en la demostración para *NP-Dureza* utilizando la técnica reducción de *3-SAT*, pero sustituyendo los bloques simples por plataformas de Donkey Kong Country.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de inicio, componente de cláusula, componente de cruce. Estos componentes se presentan en[1]. El componente de inicio solo se presenta descrito en[1], por lo que la imagen presentada aquí es una representación de esa descripción.

Componente de inicio:

Para esta demostración es necesario que el jugador pierda al tocar un solo enemigo. El componente consiste en un barril DK al inicio del nivel seguido de una pared de zings. Esta pared consiste en suficientes zings para llenar del suelo al techo. De esta manera, será necesario perder un Kong para pasar. Con esto se asegura que se recorra el escenario creado con un solo Kong y por lo tanto, se pierda una vida de un golpe al tocar un zing de la construcción completa. En la figura 2.60 se observa el componente de inicio.

Figura 2.60: Componente de inicio de Donkey Kong Country para la estructura de *3-SAT*.

Componente de cláusula:

La sección de activación del componente de cláusula se muestra en la figura 2.61. Las tres entradas superiores con los barriles se conectan con los componentes de variable de las literales de la cláusula. Al llegar por una de estas entradas, Donkey Kong puede lanzar el barril que está a su alcance hacia la derecha, para que éste ruede y mate al zinger. Este zinger es estático.

Los cañones de barril son cañones manuales que giran, los cuales le permiten tanto posicionarse en la plataforma donde se encuentra el barril, como regresar por el camino del que llegó. Si Donkey Kong intenta bajar al área donde se encuentra el zinger, no podrá regresar por ninguna entrada. De esta forma se evita que Donkey Kong intente llegar a una entrada que no sea por la que llegó.

Figura 2.61: Sección de activación de componente de cláusula de Donkey Kong Country para la estructura de *3-SAT*.

La sección de revisión de todos los componentes de cláusula se muestra en la figura 2.62. Aunque la figura no está a escala, los zingers que se muestran son los mismos zingers de cada una de las secciones de activación de las cláusulas.

Donkey Kong llega por la izquierda, y para atravesar debe tomar los barriles cañón, que son automáticos. Los barriles cañón lanzan a Donkey Kong por el camino lleno de zingers, de tal manera que si queda al menos uno vivo, el zinger matará a Donkey Kong al tocarlo.

De esta manera, Donkey Kong puede atravesar esta sección si y sólo si mató a cada zinger de la sección de activación de los componentes de cláusula. Si Donkey Kong puede pasar la sección con zingers, el cañon lo dejará en la plataforma al lado de la meta del nivel.

Figura 2.62: Sección de revisión de componente de cláusula de Donkey Kong Country para la estructura de *3-SAT*.

Componente de cruce:

El componente de cruce se muestra en la figura 2.63. Este componente es relativamente sencillo, y es bidireccional, por lo que puede utilizarse de izquierda a derecha, de derecha a izquierda, de arriba a abajo y de abajo hacia arriba. Suponiendo que se quiere atravesar de izquierda a derecha, Donkey Kong entra por la izquierda, toma el barril cañón que lo lanza a la salida derecha y ahí cae en un barril cañón que lo saca por la derecha del componente. Las otras direcciones del componente son análogas.

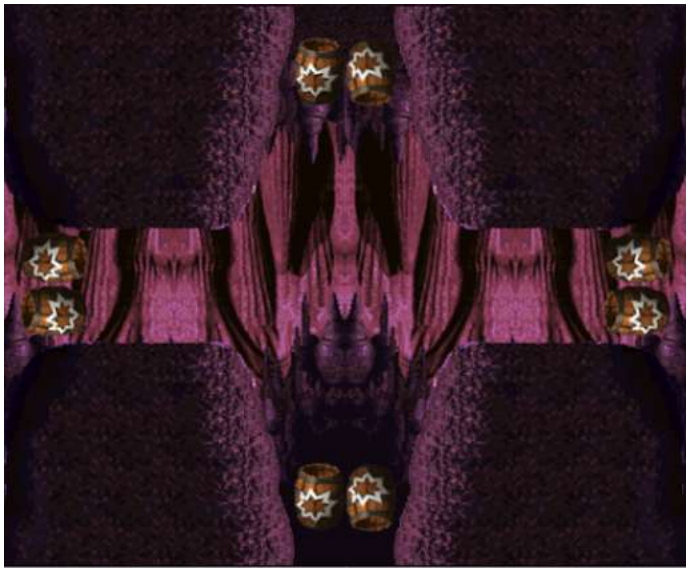


Figura 2.63: Componente de cruce de Donkey Kong Country para la estructura de *3-SAT*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si es posible terminar un nivel dado de Donkey Kong Country generalizado siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en[1] se menciona que las secuelas de este videojuego, Donkey Kong Country 2 y Donkey Kong Country 3, contienen todos los elementos utilizados para la demostración.

2.3.1.3. The Legend of Zelda: A Link to the Past (3-SAT)

En[1] se muestra la siguiente demostración de The Legend of Zelda: A Link to the Past generalizado, donde se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el problema de decisión a resolver es decidir si es posible terminar un nivel dado de The Legend of Zelda: A Link to the Past generalizado siguiendo sus reglas.

El componente de inicio no es necesario, pues no se necesita de un estado específico al recorrer el escenario. El componente de final sólo debe tener la meta del nivel.

Una vez explicada la estructura de la reducción, basta con explicar la cons-

trucción del componente de variable, componente de cláusula y componente de cruce. Estos componentes se presentan en [1].

Componente de cruce:

El componente de cruce se presenta de manera natural en el videojuego original. Como ya se explicó en las reglas del videojuego, existen plataformas a distintas elevaciones que pueden pasar una debajo de otra. Además hay escaleras que te dejan bajar a las plataformas a menor elevación.

De esta forma basta con un pasillo horizontal en una elevación superior y uno vertical en elevación inferior, las cuales pueden cruzarse sin problemas. En la figura 2.64 se observa el componente de cruce.

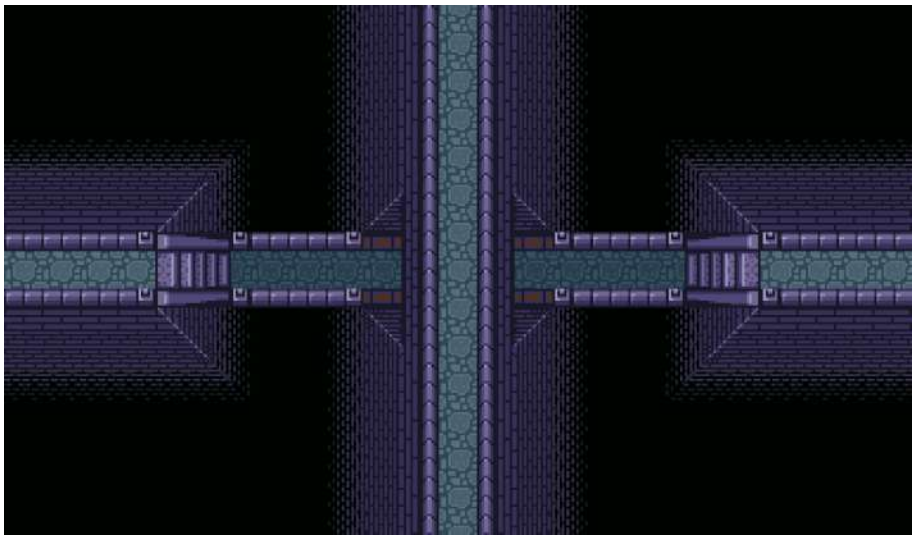


Figura 2.64: Componente de cruce de The Legend of Zelda: A Link to the Past para la estructura de *3-SAT*.

Componente de variable:

El componente de variable se muestra en la figura 2.65. Este componente se recorre de arriba a abajo y cuenta con dos entradas y dos salidas. Una salida corresponde a la asignación positiva de la variable y la otra corresponde a la asignación negativa de la variable, siendo la de la izquierda la positiva y la derecha la negativa.

Las dos entradas corresponden a las salidas del componente de variable anterior respectivamente. Para el primer componente de variable, puede

asumirse que estas dos entradas están ambas conectadas al inicio del escenario.

Link puede alcanzar el cofre central desde cualquiera de las entradas con el gancho. Una vez que Link se encuentre en la plataforma central, puede alcanzar cualquiera de los dos cofres de las salidas. Si Link elige alcanzar uno de los cofres y por lo tanto una salida, no podrá regresar a la plataforma central, y no podrá alcanzar la otra salida, por lo que no se pueden seleccionar ambas asignaciones.

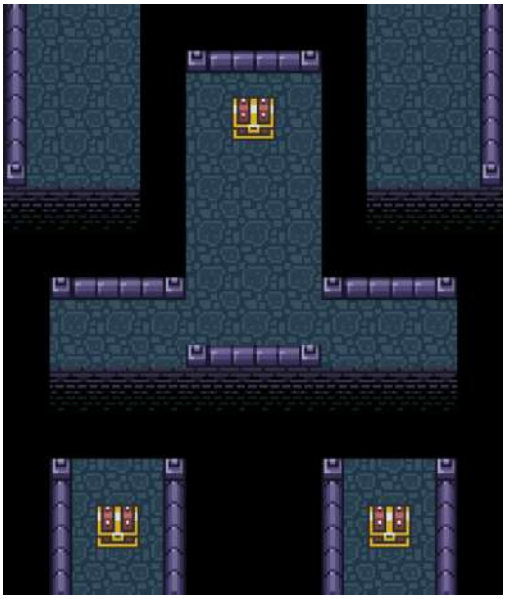


Figura 2.65: Componente de variable de The Legend of Zelda: A Link to the Past para la estructura de *3-SAT*.

Componente de cláusula:

En la figura 2.66 se observan tres componentes de cláusula juntos. La razón de mostrar tres componentes juntos, es para que la forma en que se conectan estos componentes entre sí sean más claros. La sección de activación del componente corresponde a los tres caminos superiores que se observan.

Los bloques que se muestran pueden ser empujados hacia abajo. Al llegar por uno de estos caminos desde una de las literales de la cláusula correspondiente al componente, Link puede empujar hacia abajo uno de los bloques.

En la sección de revisión, que corresponde a la parte inferior del componente, Link podrá llegar a la plataforma del siguiente componente si golpea con el gancho a uno de los bloques. Link llega a esta sección desde la izquierda. Para poder golpear al bloque, será necesario que haya sido empujado por Link hacia abajo, pues de otra forma la pared al lado del bloque le impedirá a Link golpear el bloque.

De esta forma Link podrá llegar hasta la parte de revisión de una cláusula si y sólo si empujó al menos uno de sus bloques en la sección de activación. La sección final puede encontrarse conectado directamente al último componente de cláusula, sin vacío que los separe para que el uso del gancho no sea necesario.

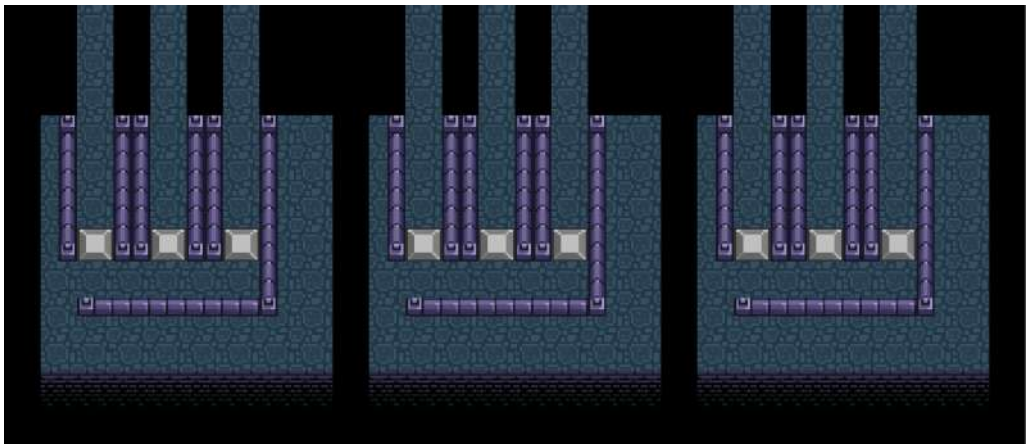


Figura 2.66: Componentes de cláusula de The Legend of Zelda: A Link to the Past para la estructura de *3-SAT*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si es posible terminar un nivel dado de The Legend of Zelda: A Link to the Past generalizado siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en[1] se menciona que algunos otros títulos de la saga que contienen los mismos elementos pueden utilizar la misma demostración. Además, se presentan demostraciones de *NP-Dureza* para otros videojuegos de la saga que no contienen los elementos utilizados o no comparten las mismas reglas. Las demostraciones de estos videojuegos dependen de los elementos y reglas de los mismos.

2.3.1.4. Metroid (3-SAT)

En[1] se muestra la siguiente demostración de Metroid generalizado, donde se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el problema de decisión a resolver es decidir si dado un escenario de Metroid generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas.

El componente de inicio no es necesario, pues no se necesita de un estado específico al recorrer el escenario. El componente de final sólo debe tener el punto final objetivo. El componente de variable es análogo al presentado para Super Mario Bros. en la demostración para *NP-Dureza* utilizando la técnica de reducción de *3-SAT*, pero sustituyendo los bloques simples por plataformas de Metroid.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de cláusula y componente de cruce. Estos componentes se presentan en[1].

Componente de cláusula:

El componente de cláusula se muestra en la figura 2.67. La sección de activación se encuentra en la parte inferior del componente. Los zoomers pueden moverse en cualquiera de las dos direcciones, pero todos se mueven en la misma dirección y a la misma velocidad. Las tres entradas corresponden a las literales de la cláusula correspondiente al componente. Al llegar por una de las entradas inferiores, Samus puede disparar hacia arriba y destruir a todos los zoomers.

Desde alguna de las tres entradas, Samus no podrá llegar a ninguna de las otras dos ni a la sección de revisión, pues sólo hay espacio de un cuadro, y Samus no puede transformarse en esfera en el aire ni saltar siendo esfera.

La sección de revisión se encuentra en la parte superior del componente. La entrada se encuentra arriba y la salida a la derecha. Para que Samus pueda pasar de la entrada a la salida por el pasillo con zoomers, es necesario que estos hayan sido destruidos anteriormente en la sección de activación.

Dado que la energía de Samus está limitada a máximo 99 puntos, siempre puede hacerse el componente lo suficientemente largo como para que Samus

no pueda pasar por el pasillo con zoomers antes de perder todos sus puntos de vida.

De esta forma, Samus puede atravesar la sección de revisión de un componente de cláusula si y sólo si destruyó a los zoomer desde una entrada de la sección de activación.

Si bien Samus puede ir a la izquierda en lugar de a la derecha, no le serviría de nada y quedará atrapada. Al igual que al entrar por la sección de activación, Samus no puede llegar desde la entrada de revisión a la sección de activación, pues hay un pasillo que debe recorrer para esto a mayor altura y con espacio de un cuadro, y Samus no puede transformarse en esfera en el aire ni saltar siendo esfera.

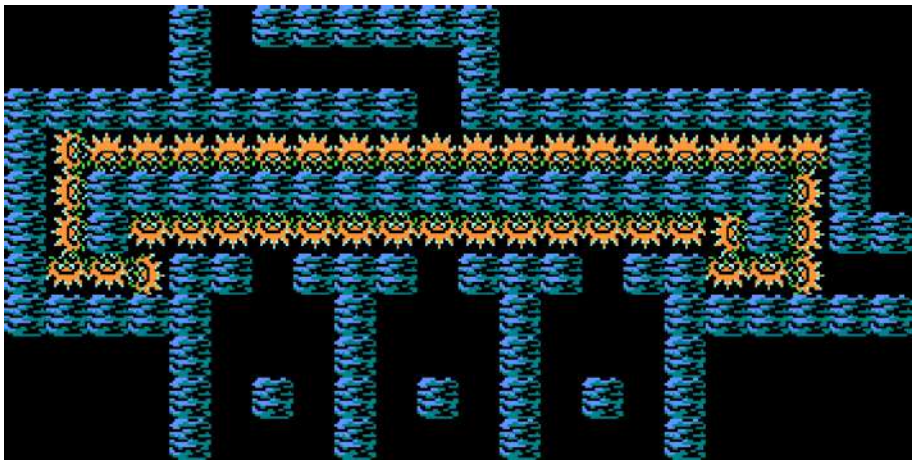


Figura 2.67: Componente de cláusula de Metroid para la estructura de *3-SAT*.

Componente de cruce:

El componente de cruce se muestra en la figura 2.68. Las tres secciones de zoomers tienen las mismas medidas y los zoomers se mueven a la misma velocidad que Samus. Las agrupaciones de zoomers se mueven en la dirección que indica la flecha cercana a estos. De esta forma, los ciclos de movimiento de los zoomers están sincronizados.

De igual forma que con el componente de cláusula, siempre puede construirse un componente con suficientes zoomers en cada agrupación para que Samus muera si trata de moverse en dirección contraria a la que se mueven los zoomers de la agrupación. Las entradas del componente se encuentran en la parte superior y las salidas por la parte inferior.

Si Samus llega por la entrada izquierda, debe esperar a que el espacio entre los zoomers se encuentre por debajo de la entrada a esta sección. De este modo, Samus puede seguir la dirección y la velocidad de los zoomers sin ser golpeada hasta caer a la siguiente sección.

Al caer, los zoomers de abajo están sincronizados de tal forma que si Samus intentara ir a la derecha, sería atacada y moriría por culpa de los Zoomers que inician abajo y en este instante se mueven hacia ella desde la derecha.

De la misma manera, los zoomers que inician arriba están sincronizados para que al caer haya suficiente tiempo para que Samus se mueva a la derecha y caiga por la salida derecha antes de que estos zoomers la golpeen.

Si Samus llega por la entrada derecha, el recorrido es análogo, por lo que Samus deberá salir por la izquierda.

Si bien Samus puede moverse a la izquierda o derecha en algún momento de sus caídas para entrar en espacios distintos a los caminos para la salida, no le serviría de nada y solo quedaría atrapada sin poder salir.

También es cierto que al caer a la sección inferior, Samus puede eliminar a los enemigos de la sección inferior. Esto último no es problema por dos cuestiones. En primera, detenerse a destruir los enemigos superiores le quitan tiempo, y ya no podría llegar a la salida sin morir. Por otro lado, el componente sólo necesita ser utilizado una vez por camino, por lo que aunque se destruyan los enemigos superiores, Samus no volverá a pasar por esa sección.

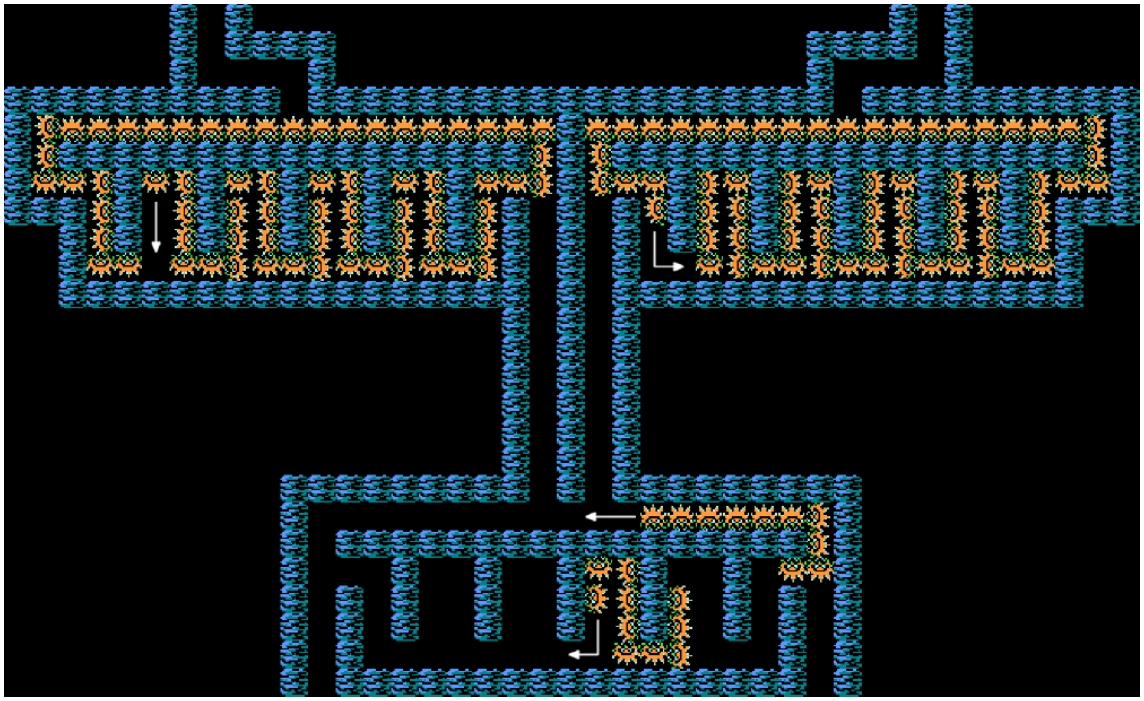


Figura 2.68: Componente de cruce de Metroid para la estructura de *3-SAT*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un escenario de Metroid generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en[1] se menciona que títulos posteriores de la saga que contienen los mismos elementos pueden utilizar la misma demostración. Sin embargo, se menciona que en videojuegos posteriores sería necesario modificar los componentes, pues en estos se permite disparar hacia abajo y cambiar de estado a esfera y de vuelta a estado normal en el aire.

2.3.1.5. Pokémon Rojo/Azul (3-SAT)

En[1] se muestra la siguiente demostración de Pokémon Rojo/Azul generalizado, donde se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el problema de decisión a resolver es decidir si dado un escenario de Pokémon Rojo/Azul generalizado, y un estado inicial del personaje, es posible llegar desde el punto inicial hasta el punto final sin perder, siguiendo sus reglas.

El componente de inicio no es necesario. Sin embargo, es necesario un estado inicial del personaje. Red debe tener en su equipo solamente un Gastly con 60 de velocidad, con el ataque autodestrucción con al menos 1 PP, el resto de los ataques con 0 PP. También es necesario que Red no tenga ningún objeto. El componente de final sólo debe tener punto final objetivo.

Para todos los componentes, se utilizarán dos tipos de entrenadores rivales, los cuales se denominan entrenador fuerte y entrenador débil. El entrenador débil tendrá en su equipo solamente un Electrode con velocidad mayor a 60 y sólo con el ataque autodestrucción. El entrenador fuerte tendrá dos snorlax, con cualquier conjunto de ataques y velocidad 30.

De esta forma, si Red entra en batalla con un entrenador débil, tendrá que usar autodestrucción, al igual que el entrenador débil. Por la velocidad, el entrenador débil atacará primero, y su pokémon se debilitará. Como el Gastly de Red es tipo fantasma y el ataque es tipo normal, Gastly no recibirá daño y la batalla terminará. Esto asegura que si Red se enfrenta a un entrenador débil, siempre ganará la batalla.

De la misma forma, si Red entra en batalla con un entrenador fuerte, tendrá que usar autodestrucción. La diferencia es que en este caso, la velocidad del pokémon de Red es mayor, por lo que atacará primero. Al atacar, los puntos de vida de su Gastly bajarán a cero y se debilitará.

Como el entrenador fuerte tiene dos pokémon, no importará que sea golpeado. Esto es porque aunque el primer pokémon del entrenador fuerte sea debilitado, aún tendrá un segundo pokémon. Como Red se quedará solamente con un pokémon derrotado, perderá la batalla. Esto asegura que si Red se enfrenta a un entrenador fuerte, siempre perderá la batalla.

En las figuras de esta demostración, los entrenadores débiles se muestran con un rango de visión rojo, mientras que los entrenadores fuertes con un rango de visión azúl.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de variable, componente de cláusula y componente de cruce. Estos componentes se presentan en [1].

Componente de variable:

El componente de variable se muestra en la figura 2.69. Red llega por la entrada a, y las salidas son b y c. La salida b corresponde a la asignación positiva de la variable y la salida c a la asignación negativa. Al llegar, Red puede realizar una de dos opciones para avanzar, caminar hacia al lado del entrenador débil y retarlo, o caminar hacia enfrente de su rango de visión y que el entrenador avance para atacar. Dado que es un entrenador débil, en ambos casos ganará la batalla.

Si Red decide atacar al entrenador por un lado, este se quedará en su posición, cerrando el camino b, pero permitiendo salir por c. Por otro lado, si Red decide caminar enfrente del rango de visión del entrenador, este caminará hacia Red y cerrará el camino c, pero abrirá el camino b. De esta manera, al elegir cualquiera de las opciones, se abre un camino pero se cierra el no elegido.

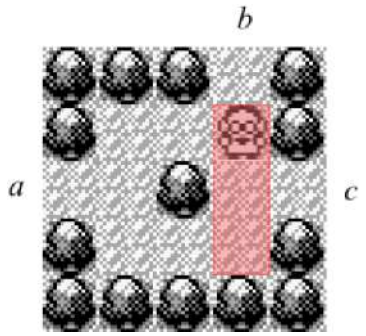


Figura 2.69: Componente de variable de Pokémon Rojo/Azul para la estructura de *3-SAT*.

Componente de cláusula:

El componente de cláusula se muestra en la figura 2.70. La sección de activación se encuentra en la parte superior del componente. Las tres entradas corresponden a las literales de la cláusula correspondiente al componente. Al llegar por una de estas entradas, Red puede activar al entrenador débil debajo de él. Si lo activa, ganará la batalla y el entrenador se quedará en su lugar en estado enfrentado.

La sección de revisión se encuentra en la parte inferior del componente. Red llega por la entrada de la izquierda. Los entrenadores débiles en estado no enfrentado de arriba caminarán hacia Red cuando pase por enfrente de su

rango de visión. Si todos los entrenadores débiles bajan liberarán el rango de visión del entrenador fuerte con dirección de visión hacia la derecha.

Si el rango de visión de ese entrenador está libre, atacará a Red cuando pase por este rango cuando intente alcanzar la salida. Como es un entrenador fuerte, Red perderá. La única forma de que esto no suceda es que al menos uno de los entrenadores débiles se encuentre en estado de enfrentado y no baje, bloqueando el rango de visión del entrenador fuerte.

De esta manera, Red puede atravesar la sección de revisión del componente si y sólo si activó una batalla con un entrenador débil desde una entrada de la sección de activación. El entrenador fuerte con dirección de visión hacia arriba se encuentra ahí para evitar que Red intente llegar desde la sección de revisión a las entradas de la sección de activación.

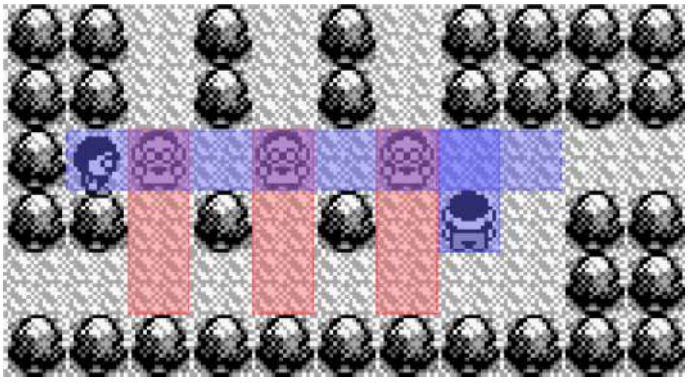


Figura 2.70: Componente de cláusula de Pokémon Rojo/Azul para la estructura de *3-SAT*.

Componente de cruce:

Para explicar este componente de manera más clara, será útil explicar un componente conocido como camino de un solo uso. Como su nombre lo indica, un camino de un sólo uso es aquel que se puede recorrer en una sola dirección una única vez.

El camino de un solo uso se muestra en la figura 2.71. En este componente, Red llega por la entrada a y debe llegar a la salida b. La primera vez que recorre el componente, Red puede llegar hasta el rango de visión del entrenador débil sin ser atacado antes, pues los rangos de visión de los entrenadores fuertes están bloqueados. Red no puede activar al entrenador débil desde arriba porque lo atacaría un entrenador fuerte.

Al llegar al rango de visión del entrenador débil, este se moverá hacia Red. Esto liberará el rango de visión del entrenador fuerte con dirección de visión hacia arriba. Al liberar el rango de visión de este entrenador, no será posible recorrer el camino de nuevo sin que este ataque a Red y lo derrote.

Si Red intenta recorrer el camino de b a a, liberará el rango de visión del entrenador fuerte cuando entre en el rango de visión del entrenador débil, por lo que no podrá llegar a a.

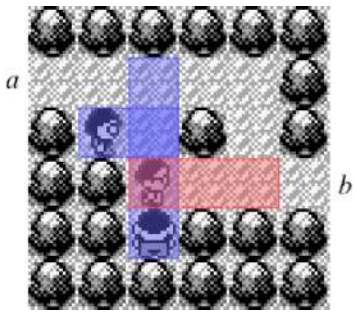


Figura 2.71: Camino de un solo uso de Pokémon Rojo/Azul para la estructura de *3-SAT*.

Una vez explicado el componente anterior, se procederá a explicar el componente de cruce. Este componente se muestra en la figura 2.72. En este componente, los caminos que se deben recorrer son de x a x' y de y a y'.

Para recorrer el camino de x a x', Red debe activar al entrenador débil que se encuentra en el camino de debajo de él, en caso de que se encuentre en estado de no enfrentado. De no hacerlo, Red no podrá avanzar posteriormente, pero esto se explicará más adelante. Una vez que lo haya activado, el único camino que puede seguir es el del camino de un sólo uso a la derecha de la entrada. Al atravesar este camino, llegará a una parte con tres caminos a seguir.

El de arriba conecta con una salida de un camino de un solo uso, por lo que no podrá avanzar por ahí. El de abajo, le permitirá atravesar un camino de un solo uso, pero no podrá llegar hasta la salida y' sin ser atrapado por el entrenador débil de rango de visión grande de hasta arriba. Por lo tanto, debe elegir el camino de la derecha al salir del primer camino de un solo uso si quiere seguir avanzando.

Al seguir por esta ruta, atravesará otro camino de un solo uso, lo que no le permitirá regresar. Terminando este camino de un solo uso, será atacado por el entrenador débil con rango de visión grande que se encuentra en la parte superior, lo cual lo dejará por encima de él en estado enfrentado.

Finalmente podrá llegar hasta la salida x'. Si el entrenador débil que se encuentra debajo de la entrada x se encuentra en estado no enfrentado, caminará hacia Red y bloqueará su camino antes de llegar a la salida x'.

El camino de y a y' es análogo al camino de x a x'.

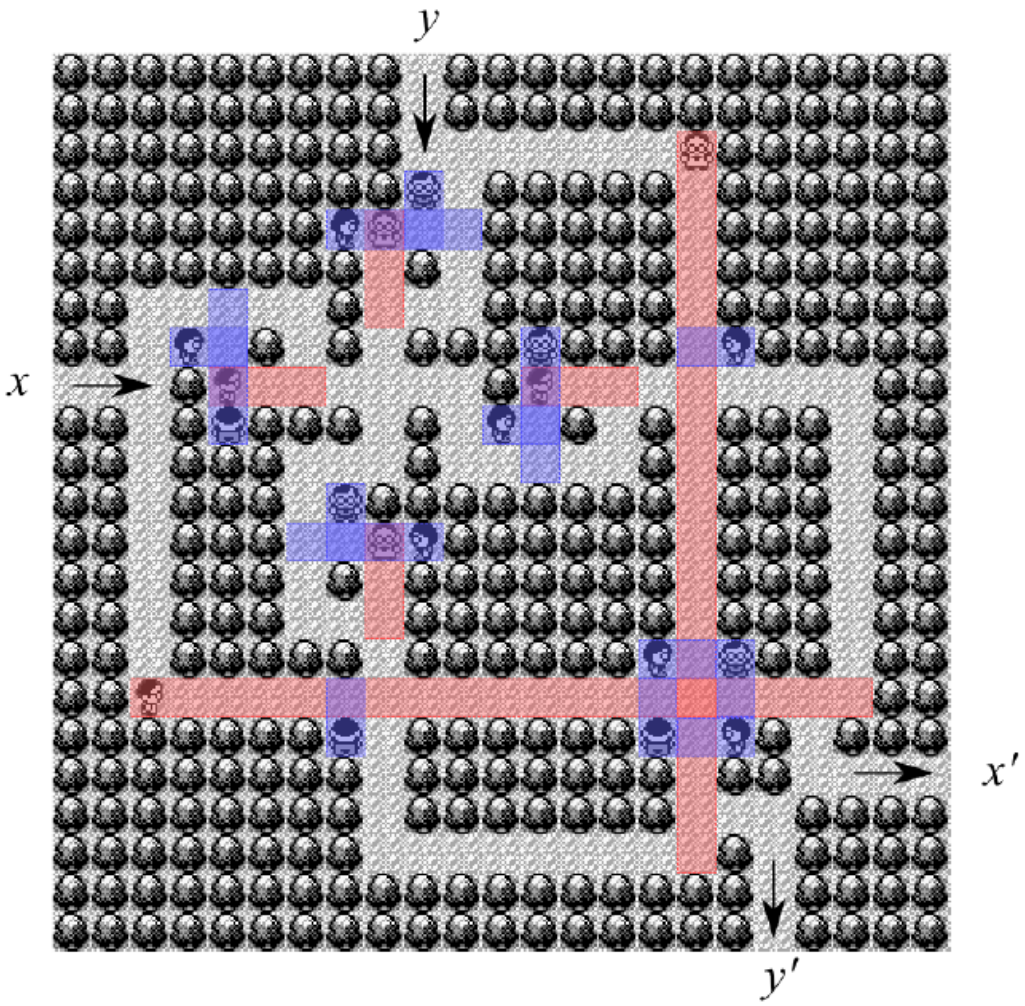


Figura 2.72: Componente de cruce de Pokémon Rojo/Azul para la estructura de *3-SAT*.

A partir de esta construcción, y utilizando la téncnica mencionada, se puede concluir que decidir si dado un escenario de Pokémon Rojo/Azul genera-

lizado, y un estado inicial del personaje, es posible llegar desde el punto inicial hasta el punto final sin perder, siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en[1] se menciona que títulos posteriores de la saga que comparte algunas reglas pueden utilizar la misma demostración. Sin embargo, se menciona que en videojuegos futuros sería necesario utilizar otros pokémon para los entrenadores y para el personaje principal en la demostración.

2.3.1.6. Mario Kart 64 (3-SAT)

En[4] se muestra la siguiente demostración de Mario Kart 64 generalizado, donde se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el problema de decisión a resolver es decidir si dado un escenario de Mario Kart 64 generalizado, un estado inicial del escenario y un tiempo dado, es posible dar una vuelta al escenario, partiendo del estado inicial en el tiempo dado en modo contrarreloj, siguiendo sus reglas.

El componente de inicio no es necesario, pero si es necesario un estado inicial del escenario. Dado que los escenarios no contienen plátanos en un inicio, se supondrá que el videojuego se ha jugado durante un tiempo y se han dejado los plátanos en los lugares mostrados en los componentes. Esto es parte del estado inicial del problema. El componente de final sólo debe tener la meta, que de hecho también es el inicio.

A diferencia de otras demostraciones, esta depende de un parámetro extra, el tiempo. El escenario general de esta reducción estará construido de tal manera que es posible recorrerlo todo sin detenerse y si no se es golpeado por ningún objeto. Sin embargo, si el jugador es golpeado con al menos un objeto, el tiempo necesario para terminarlo será estrictamente mayor al pedido.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de variable, componente de cláusula y componente de cruce. Estos componentes se presentan en[4].

Componente de variable:

El componente de variable se compone de una bifurcación. Cada uno de los caminos se encuentra conectado a las secciones de activación de los compo-

nentes de las cláusulas que los contienen como literal. Los componentes de cláusulas que contienen a la variable positiva tienen conexión con el camino izquierdo, mientras que los componentes de cláusulas que contienen a la variable negada tienen conexión con el camino derecho.

Ambos caminos tienen el mismo tiempo óptimo para ser recorridos. Es seguro suponer que el jugador no volverá por un camino una vez seleccionado, pues esto le tomaría tiempo y ya no le sería posible terminar en el tiempo requerido.

En la figura 2.73 se observa el componente de variable. Como puede observarse, en cada camino se encuentran cajas de objetos con caparazones verdes. Tomar y usar estos caparazones no retrasa al jugador. El uso de estos caparazones será explicado con mayor detalle en el componente de cláusula.

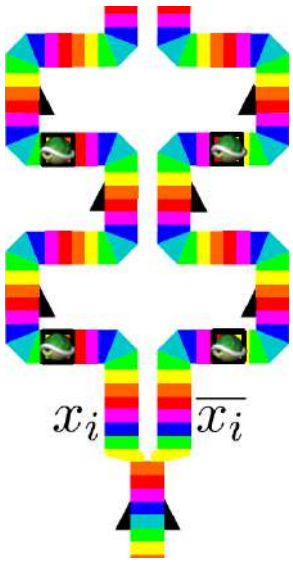


Figura 2.73: Componente de variable de Mario Kart 64 para la estructura de *3-SAT*.

Componente de cláusula:

Antes de explicar a detalle este componente, es importante que el jugador llegue hasta el inicio de las secciones de revisión de la estructura sin caparazones. Para lograr esto, al inicio de este camino se encuentra el componente de limpieza mostrado en la figura 2.74. En este componente se tienen dos

cajas de objetos con caparazones verdes seguidos de dos plátanos en el suelo.

Los plátanos se encuentran a una distancia de los caparazones mayor a la que pueden recorrer estos últimos antes de desaparecer.

De esta forma, la única forma en que se pueden destruir ambos plátanos, será mantener un caparazón y tomar un segundo en el espacio de objetos. De esta manera se podrán destruir ambos plátanos sin detenerse ni ser golpeado por ellos. De esta forma, al avanzar más allá de los plátanos, el jugador no tendrá ningún caparazón.

Figura 2.74: Componente de limpieza al inicio de camino de revisión de Mario Kart 64 para la estructura de *3-SAT*.

Una vez explicado el componente anterior, se procederá a explicar el componente de cláusula. La sección de revisión se muestra en la figura 2.75. Los tres caminos que se muestran tienen el mismo tiempo óptimo para ser recorridos. Estos caminos corresponden a las literales de la cláusula del componente.

Como se puede observar, en todos los caminos se encuentra un plátano en el suelo. Como se dijo al inicio, para poder completar la pista en el tiempo requerido, el jugador no debe chocar con ningún objeto. Por lo tanto, es necesario que alguno de los caminos no tenga el plátano cuando el jugador pase por ahí. La forma de quitar estos plátanos se explicará en la sección de activación.

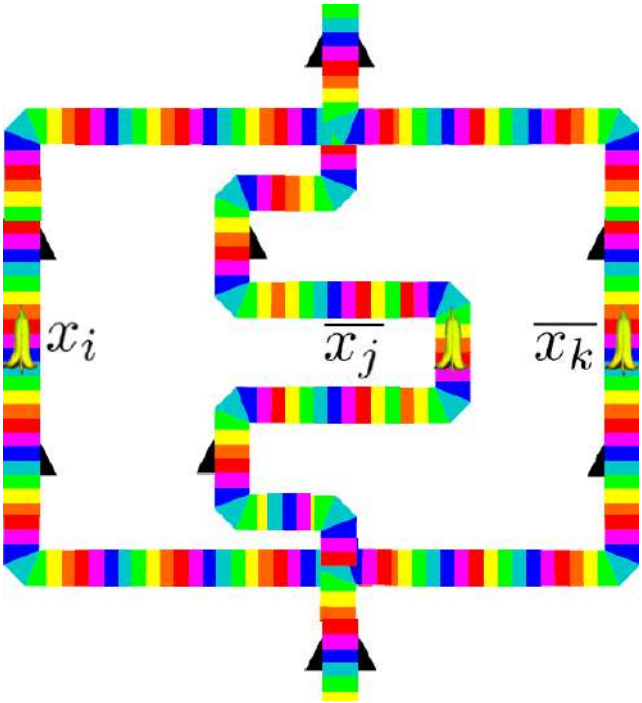


Figura 2.75: Sección de revisión del componente de variable de Mario Kart 64 para la estructura de *3-SAT*.

En la figura 2.76 se observan las secciones de activación de dos componentes de cláusula que contienen la misma literal. La parte sombreada representa que esta plataforma se encuentra en un nivel inferior.

Los caparazones, que son los mismos que podían observarse desde el componente de variable, pueden ser lanzados hacia abajo para destruir los plátanos.

Cada uno de los tres caminos de la sección de revisión tienen esta forma. De esta forma, será posible pasar por la sección de revisión sin golpear ningun plátano si y sólo si, se eligió al menos uno de los caminos de las literales de la cláusula.

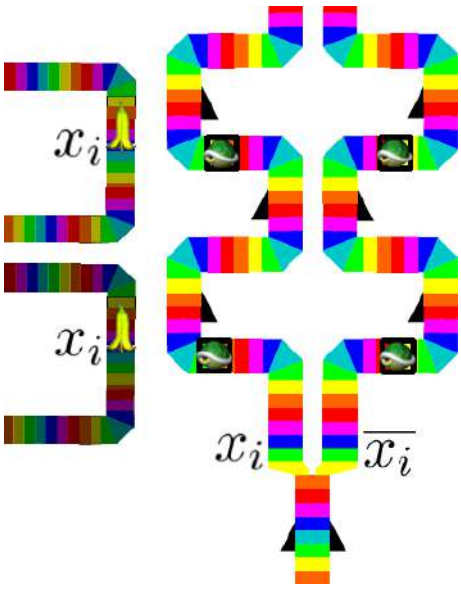


Figura 2.76: Sección de activación del componente de variable de Mario Kart 64 para la estructura de *3-SAT*.

Componente de cruce:

El componente de cruce es sencillo. Como se explica en las reglas, el escenario permite que haya plataformas a distintas elevaciones. Además, puede haber superficies inclinadas que te permiten moverte a plataformas de menor elevación. Por lo tanto, es posible crear un cruce con dos pasillos a distinta elevación.

Para evitar que el jugador baje a la otra plataforma o lance un caparazón, se agrega una barda en las orillas de ambas plataformas. El componente de cruce se muestra en la figura 2.77.

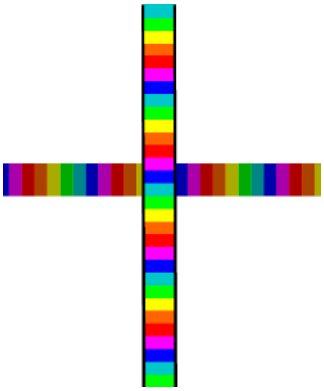


Figura 2.77: Componente de cruce de Mario Kart 64 para la estructura de *3-SAT*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un estado escenario de Mario Kart 64 generalizado, un estado inicial del escenario y un tiempo dado, es posible dar una vuelta al escenario, partiendo del estado inicial en el tiempo dado en modo contrarreloj, siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en[4] se menciona que esta demostración puede ser utilizada para todos los videojuegos de la misma saga que sean 3D. Sin embargo, el último videojuego de la saga no permite mantener objetos, por lo que el componente que utiliza este hecho es ligeramente distinto, con solo un caparazón y un plátano. Los videojuegos que no son en 3D no cuentan con plataformas a distintas elevaciones, por lo que no se puede aplicar la demostración en estos.

2.3.1.7. Portal (3-SAT)

En[8] se muestra la siguiente demostración de Portal generalizado, donde se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el problema de decisión a resolver es decidir si dado un escenario de Portal generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas.

El componente de inicio no es necesario, pues no se necesita de un estado específico al recorrer el escenario. El componente de final sólo debe tener el punto final objetivo. El componente de cruce tampoco es necesario, pues

al ser un escenario en 3D con plataformas a distinta elevación, basta con crear dos pasillos cerrados a distintas elevaciones para los cruces.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de variable y componente de cláusula. Estos componentes se presentan en[8].

Componente de variable:

El componente de variable es sencillo. Consiste en una sección del nivel con una entrada en la parte superior. Posteriormente que se divide a dos caminos disjuntos. Cada uno de estos caminos se conectan a una superficie con menor elevación que el camino de donde llega el jugador, con una caída larga. Este componente puede verse en la figura 2.78.

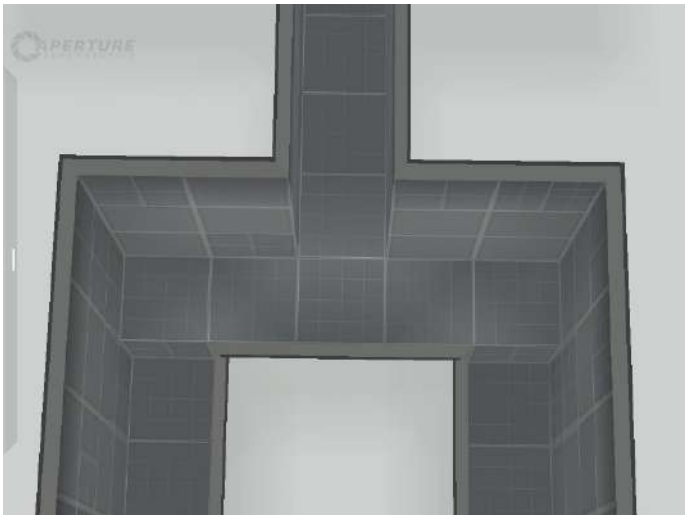


Figura 2.78: Componente de variable de Portal para la estructura de *3-SAT*.

Componente de cláusula:

Para mayor claridad en este componente, lo primero que se hará es explicar cómo se compone cada sección de literal del componente. Para cada literal, se tendrá una estructura como la mostrada en la figura 2.79.

Cada literal consiste en su propia sección de activación y revisión. La sección de activación se encuentra a la derecha de la imagen y cuenta con una entrada y una salida. Estos dos caminos, como en las otras demostraciones, se conectan al camino de la literal correspondiente.

Desde esta sección, se tiene acceso a la parte trasera de tres torretas, fuera de su campo de visión. De esta manera, desde esta sección el jugador puede desactivar a las torretas sin recibir daño.

Por otro lado, la sección de revisión se tiene un camino desde el cual no se tiene acceso directo a las torretas ni a la sección de activación en general. Para evitar este acceso se tiene una caída larga entre las torretas y el camino.

Esta caída larga tiene una distancia mayor a la que Chell puede saltar. Sin embargo, este camino está en el campo de visión de las torretas. Se tiene la cantidad suficiente de torretas para que al intentar pasar, Chell muera por los disparos de las mismas.

De esta forma, la única forma de atravesar la sección de revisión es haber desactivado las torretas desde la sección de activación.

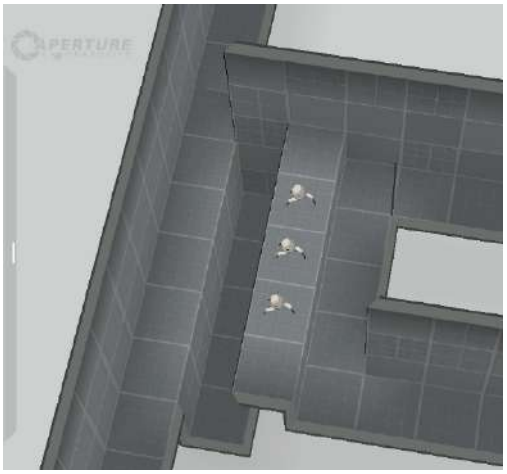


Figura 2.79: Sección de una literal del componente de cláusula de Portal para la estructura de *3-SAT*.

Una vez explicada cada literal, solo haría falta explicar cómo se conectan en el componente. Para el componente, se tienen una entrada y una salida, separadas por tres pasillos. En cada pasillo se encuentra la sección de la literal correspondiente a la cláusula del componente. Estos pasillos pueden encontrarse a distintas alturas, conectados con plataformas a distintas alturas para que el jugador pueda acceder a los mismos.

Es necesario que algunos caminos se encuentren a distintas alturas y unidos

con escaleras para que las salidas y entradas de las secciones de literales no choque con los pasillos. En la figura 2.80 se muestra el componente de cláusula. Esta figura se obtuvo basándose en la presentada en [8].

De esta forma, será posible pasar por la sección de revisión del componente de cláusula si y sólo si, se eligió al menos uno de los caminos de las literales de la cláusula en la sección de activación, y se desactivaron las torretas del mismo.

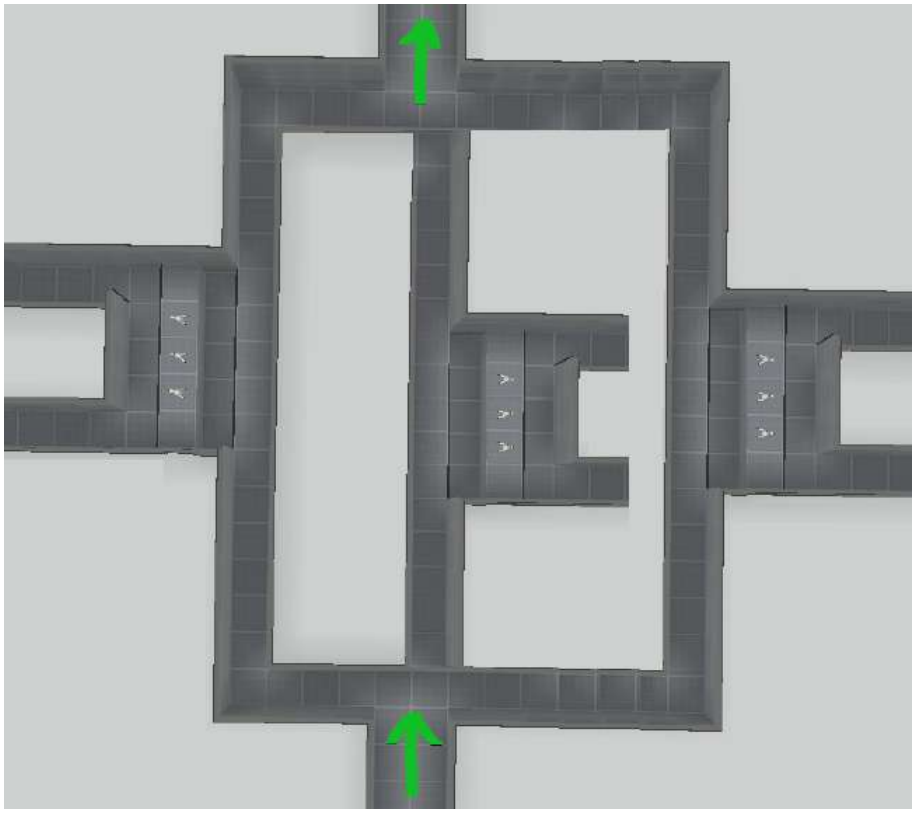


Figura 2.80: Componente de cláusula de Portal para la estructura de *3-SAT*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un escenario de Portal generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en [8] se menciona que esta demostración puede extenderse fácilmente a muchos otros videojuegos con los mismos elementos. En principio, funciona para la secuela de este videojuego, Portal 2. Sin embargo,

otros videojuegos, principalmente de sigilo o disparos, pueden utilizar una demostración similar a esta.

Solo se necesita que los videojuegos se lleven a cabo en escenarios 3D, y que puedan implementarse caídas largas y enemigos que puedan ser derrotados o puestos fuera de combate desde un área distinta al área en la que el personaje recibe daño de ellos.

2.3.1.8. Portal (Ciclo hamiltoniano)

Como se muestra en[8], Portal generalizado cuenta con los elementos necesarios para utilizar la técnica para *NP-Dureza* de reducción de ciclo hamiltoniano en una gráfica en cuadrícula.

En este caso, el problema de decisión a resolver es decidir si dado un nivel de Portal generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas.

Para cada nodo de la gráfica, se utilizará un cuarto vacío con un botón temporal en el centro. Estos cuartos pueden verse en la figura 2.81.

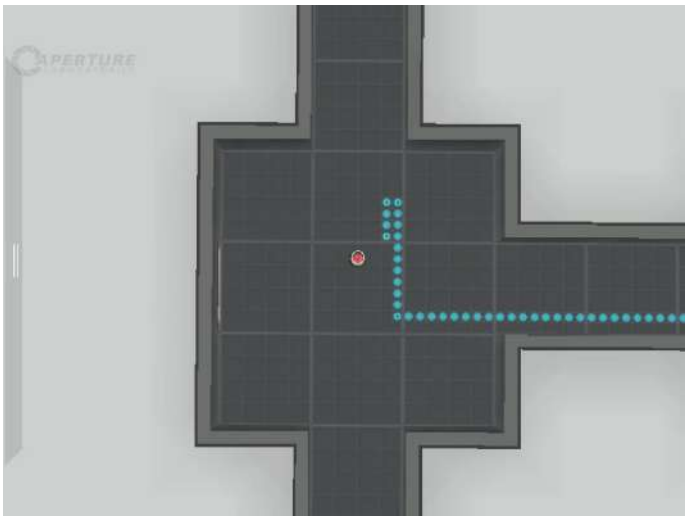


Figura 2.81: Cuarto de Portal correspondiente a un nodo de la gráfica en cuadrícula para la estructura de ciclo hamiltoniano.

Por otro lado, en[8] se menciona que las puertas de la estructura de la

técnica pueden ser implementadas por cualquier elemento que bloquee el paso de Chell. En este caso se utilizan escaleras retráctiles.

Estas escaleras se posicionan una tras otra, pero cada una se encuentra en una posición ligeramente más alta que la anterior, de tal manera que la diferencia de altura entre una escalera y la siguiente es la suficiente para que solo sea posible alcanzarla si la escalera anterior se encuentra activada. De esta forma, la única forma de atravesar la secuencia de escaleras, es si todas están activadas.

En la figura 2.82 se observa esta secuencia de escaleras retráctiles, con solo algunas activadas.

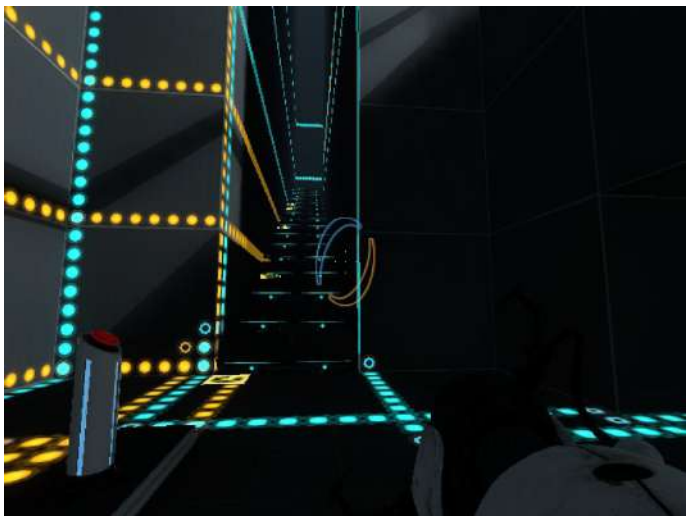


Figura 2.82: Secuencia de escaleras retráctiles en Portal que funcionan como puertas para la estructura de ciclo hamiltoniano.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un escenario de Portal generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en[8] se menciona que la secuela Portal 2 cuenta con todos los elementos necesarios para aplicar las misma demostración.

2.3.2. NP-Dureza sin técnica

2.3.2.1. Super Mario Bros. (Mochila 1-0)

Para esta demostración de Super Mario Bros. generalizado presentada en [9], se realizará una reducción del problema de la mochila 0-1. Este problema puede encontrarse en los 21 problemas originales de [15] junto con su demostración de *NP-Completitud*. Este problema es el siguiente:

Entrada: Un conjunto de elementos U, para cada $u \in U$ un peso $w(u) \in \mathbb{Z}^+$ y valor $v(u) \in \mathbb{Z}^+$, una capacidad máxima de la mochila $W \in \mathbb{Z}^+$ y un valor objetivo $V \in \mathbb{Z}^+$.

Problema de decisión: Decidir si existe un subconjunto $U' \subseteq U$ de elementos con peso total $\sum w(u) \leq W$ y valor $\sum v(u) \geq V$.

Este problema es débilmente *NP-Duro*, pues existen algoritmos pseudo polinomiales para resolverlo. El problema se presenta en [11] como *NP-Completo* incluso si el peso de cada elemento es proporcional a su valor.

En [9] se muestra que el problema continúa siendo *NP-Completo* incluso si los pesos se restringen a $w(u) \geq 100 * v(u)$ para todo $u \in U$. En esta reducción se utilizarán instancias con estas características.

Para esta reducción se utiliza un tamaño no acotado para la longitud del nivel. Sin embargo, no se necesita que la altura del nivel sea no acotada. De hecho sólo se requiere una altura de dos cuadros.

De igual manera, la restricción del tamaño de la pantalla visible no se utiliza en la demostración, por lo que no es necesario que esta restricción se omita. Dado que no se utilizan enemigos ni objetos, tampoco se necesita quitar la restricción de elementos que se reinician al salir de la pantalla.

En general, sólo se utilizarán dos controles, moverse a la derecha y agacharse para entrar en tuberías.

Para la reducción se utilizaráuna versión de Super Mario Bros. con una codificación en la que varias copias consecutivas de un mismo elemento o nivel pueden codificarse utilizando $O(\log m)$ bits, donde m es el número de repeticiones del elemento o nivel. Esta versión se define en [9].

En esta reducción se tiene una instancia del problema de la mochila 0-1 con las características explicadas anteriormente, y se reducirá a una serie de niveles de Super Mario Bros.

En este caso el problema de decisión a resolver es decidir si es posible completar una serie de niveles dados de Super Mario Bros. generalizado, siguiendo sus reglas.

Para esta reducción se utilizarán tres tipos de niveles. Estos niveles serán llamados nivel de decisión, nivel de monedas y nivel de muerte.

Nivel de decisión:

El nivel de decisión se compone de $n + 3$ secciones.

La primera sección inicia con una pantalla vacía, en la cual al dar un paso se activa un punto de control. Si Mario muere y se reinicia al punto de control, el límite de tiempo se establecerá en 100.

La segunda sección es una sección vacía. Esta sección requiere 100 unidades de tiempo para ser recorrida.

Seguido de estas dos secciones, hay n secciones de decisión. La sección i, correspondiente al elemento $u_i \in U$, inicia con una tubería de entrada y terminan con una tubería de salida, conectadas entre sí. Al entrar a la tubería del principio Mario saldrá por la tubería del final, saltándose toda la sección. De no entrar, Mario tendrá que recorrer la sección a pie y gastar una cantidad de tiempo.

En la sección se encuentran $v(u_i)$ monedas y requiere $w(u_i)$ unidades de tiempo para atravesarse. Estas monedas se encuentran separadas de las tuberías, de tal manera que Mario deberá decidir si tomar las monedas con el costo de tiempo o saltar la sección sin tomar ninguna moneda.

La sección final cuenta con una meta para el nivel.

El límite de tiempo inicial del nivel es $W + E$, donde E es el tiempo que toma terminar el nivel utilizando los n tubos. Este valor E incluye las 100 unidades de tiempo de la primera sección del nivel. De esta manera, Mario está obligado a terminar el nivel con una sola vida, ya que si muere, sólo

tendrá 100 unidades para terminar el nivel, lo cual no es suficiente. El nivel de decisión se muestra en la figura 2.83.

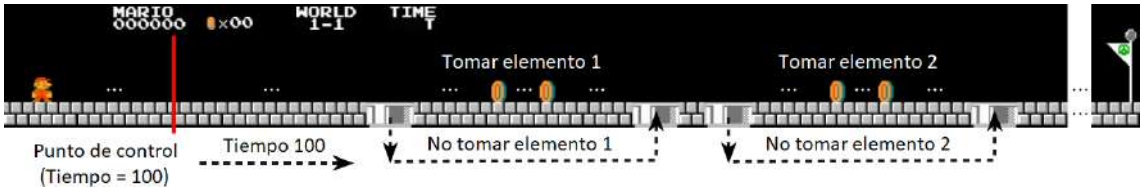


Figura 2.83: Nivel de decisión para la reducción del problema de la mochila a Super Mario Bros.

Nivel de monedas:

Un nivel de monedas contiene c monedas, para una c que cumple $0 \leq c \leq 99$, $c \in \mathbb{Z}^+$. En[9] se asignan 200 unidades de tiempo y se menciona que son suficientes para recorrerlo. De no ser así, la cantidad de tiempo para recorrerlo sigue siendo a lo más una constante, por lo que al límite de tiempo se le asigna el valor de esta constante. El nivel de monedas se muestra en la figura 2.84.

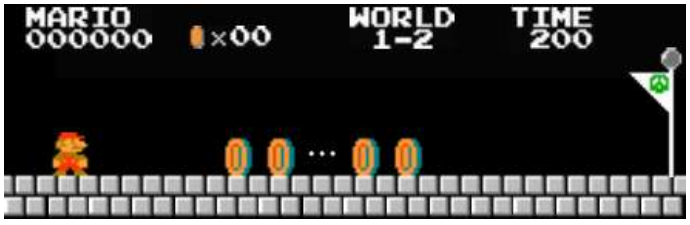


Figura 2.84: Nivel de monedas para la reducción del problema de la mochila a Super Mario Bros.

Nivel de muerte:

Un nivel de muerte es un nivel vacío con una meta al final que requiere de 300 unidades de tiempo para ser recorrido. Sin embargo, el límite de tiempo inicial es de 200 unidades. En la mitad del nivel se encuentra un punto de control. Al reaparecer en el punto de control, el límite de tiempo se establece en 200.

De esta manera, el nivel puede completarse si Mario llega hasta el punto

de control, muere y reaparece con suficiente tiempo para llegar al final del nivel. El nivel de muerte se muestra en la figura 2.85.

Figura 2.85: Nivel de muerte para la reducción del problema de la mochila a Super Mario Bros.

Una vez explicados los niveles, se procederá a explicar la configuración de la sucesión de niveles.

Sea una instancia del problema de la mochila 0-1 que satisface la definición y la restricción de que los pesos cumplen que $w(u) \geq 100 \cdot v(u)$ para todo $u \in U$. Sea $L = \lceil V/100 \rceil$.

El primer nivel consiste en un nivel de decisión donde cada sección de decisión corresponde a un elemento de $u \in U$, y el límite de tiempo es $W + E$.

El segundo nivel consiste en un nivel de monedas con $c = 100 \cdot L - V$.

Los siguientes $L+2$ niveles consisten en niveles de muerte. En estos niveles es donde será importante el uso de la codificación definida anteriormente, ya que el número de niveles puede crecer exponencialmente.

Hay que notar que Mario podrá atravesar los niveles de muerte si y sólo si tiene al menos $L + 3$ vidas al llegar al primer nivel de muerte. Dado que Mario inicia el videojuego con 3 vidas, Mario debe tomar al menos $V + c$ monedas para obtener las vidas necesarias.

Como $c = 100 \cdot L - V$, hace falta tomar V monedas en el primer nivel. Por la construcción, Mario puede tomar V monedas en el primer nivel si y sólo si la instancia del problema de la mochila 0-1 tiene solución.

Por lo tanto, es posible terminar una serie de niveles en Super Mario Bros.

generalizado si y sólo si la instancia del problema de la mochila 0-1 tiene solución.

Por otro lado, el nivel de decisión se puede construir en tiempo y espacio polinomial. por la configuración descrita para guardar las copias de elementos en los niveles, la descripción del nivel de decisión puede construir en tiempo lineal con respecto a la entrada. Por la misma configuración, las copias de niveles de muerte pueden guardarse en tamaño lineal con respecto a la entrada. De esta manera, la sucesión de niveles puede construirse en tiempo y espacio polinomial.

Por lo tanto, el problema es *NP-Duro*.

Adicionalmente, en[9] se muestra que es posible realizar la reducción sin utilizar la suposición de la codificación de niveles consecutivos en $O(\log m)$. Para esto es necesario suponer que la cantidad de monedas que otorgan un vida a Mario puede configurarse a voluntad.

2.3.2.2. Tetris (3-Partición)

Para esta demostración de Tetris presentada en[5], se realizará una reducción del problema 3-Partición. Este problema se encuentra demostrado como *NP-Completo* en[11]. Este problema es el siguiente:

Entrada: Un conjunto $U = \{a1, ..., a_{3 \cdot s}\}$ de enteros no negativos y un entero no negativo T, tal que $T/4 < a_i < T/2$ para toda i que cumple $1 \leq i \leq 3 \cdot s$ y $\sum_{i=1}^{3 \cdot s} a_i = s \cdot T$.

Problema de decisión: Decidir si existe una partición de U en s subconjuntos $A_1, ..., A_s$ tal que para todo $1 \leq j \leq s$ se tenga que $\sum_{a_i \in A_j} a_i = T$.

En[5] se muestra que el problema continúa siendo *NP-Completo* incluso si se limita a instancias con las siguientes condiciones:

- Para cualquier conjunto $S \subseteq U$, si $\sum_{a_i \in S} a_i = T$ entonces $|S| = 3$.
- T es par.

- Si $\sum_{a_i \in A_j} a_i \neq T$, entonces $|T - \sum_{a_i \in A_j} a_i| \geq 3 \cdot s$.

En esta reducción se utilizarán instancias con estas características.

En esta reducción se tiene una instancia del problema de 3-Partición, y se reducirá a un tablero inicial de Tetris con una secuencia de piezas.

En este caso el problema de decisión a resolver es decidir si es posible limpiar la pantalla dada una configuración inicial de escenario de Tetris generalizado y una secuencia de piezas, siguiendo sus reglas.

Para explicar la construcción, primero se explicará el tablero inicial y posteriormente la secuencia de piezas.

Tablero inicial:

El tablero inicial se compone de $6 \cdot T + 22 + (3 \cdot s + O(1))$ filas y $6 \cdot s + 3$ columnas. Intuitivamente, las primeras $6 \cdot s$ columnas corresponden a los s conjuntos, por lo que a cada uno le corresponden seis columnas. Las otras tres columnas corresponden a columnas de control para que no puedan destruirse filas hasta el final de la construcción.

Por otro lado, las primeras $6 \cdot T + 22$ filas corresponden al espacio que debe llenarse por cada conjunto. Las primeras cuatro filas corresponden a filas de control que aseguran que las piezas se posicionan de manera correcta.

Posteriormente, las siguientes $6 \cdot T + 18$ filas tienen el propósito de acomodar las piezas de los elementos de cada conjunto. A cada número a_i le corresponderá un espacio de $a_i + 1$ bloques de piezas que ocupan seis filas.

Dado que la suma de los tres números del conjunto $A_j = \{a_i, a_j, a_k\}$ debe ser T, al sumar las filas de las piezas resulta $6 \cdot (T + 3) = 6 \cdot T + 18$. Las $3 \cdot s + O(1)$ filas finales corresponden a espacio suficiente para que el jugador pueda mover y rotar las piezas a cualquier posición horizontal y cualquier orientación de pieza.

Estas últimas filas se encontrarán vacías por completo. La decisión de las $3 \cdot s + O(1)$ filas vacías dependen de un modelo realista de las capacidades de un jugador para mover las piezas, y se explican con más detalle en [5].

El tablero inicial se puede observar en la figura 2.86. Cada una de las secciones que corresponden a los s conjuntos se componen de las siguientes columnas, considerando que al decir que una columna está completamente llena, se refiere a las primeras $6 \cdot T + 22$ filas:

- Las primeras dos columnas están vacías, excepto por las primeras cuatro filas de cada una.
- La tercera columna está completamente vacía.
- La cuarta y quinta columnas están llena en todas las filas $h \neq 5$ (mód 6) y vacías en las filas $h \equiv 5$ (mód 6).
- La sexta columna está completamente llena.

Para explicar con mayor claridad, se le llamará a las primeras cuatro columnas de cada una de estas secciones, cubetas.

Por otro lado, la sección de control se compone de las siguientes columnas, considerando de nuevo sólo las primeras $6 \cdot T + 22$ filas:

- La primera columna está llena excepto por las últimas dos filas.
- La segunda columna está llena excepto por la última fila.
- La tercera columna está vacía excepto por la penúltima fila.

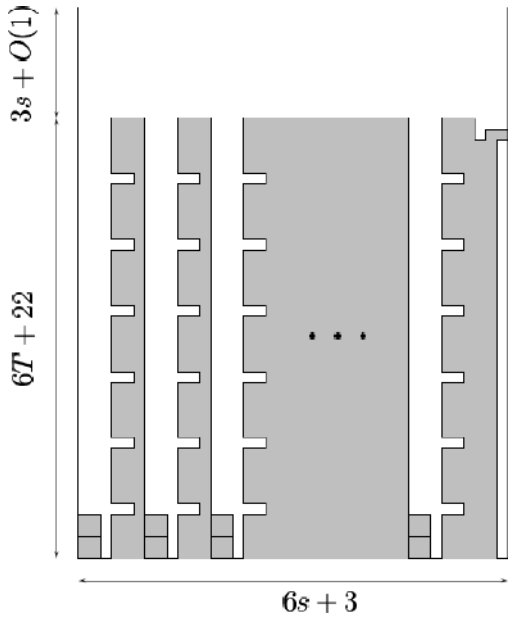


Figura 2.86: Tablero inicial de Tetris para la reducción de del problema 3-partición.

Piezas:

La secuencia de piezas se divide en dos partes. La primera parte corresponde a los elementos de U. La segunda parte corresponde a una secuencia de control que permite limpiar el escenario.

Por cada elemento $a_i \in U$, se tienen las siguientes piezas en la secuencia:

- Una sección de inicio, que corresponde a la secuencia $\{l, Li, C\}$.
- Una sección para rellenar, que consiste en la secuencia $\{Li, Si, Li, Li, C\}$ repetida a_i veces.
- Una sección para finalizar, que consiste en la secuencia $\{C, C\}$.

En la figura 2.87 se observa la forma en que se espera que la secuencia de piezas de cada elemento se acomoden en el escenario.

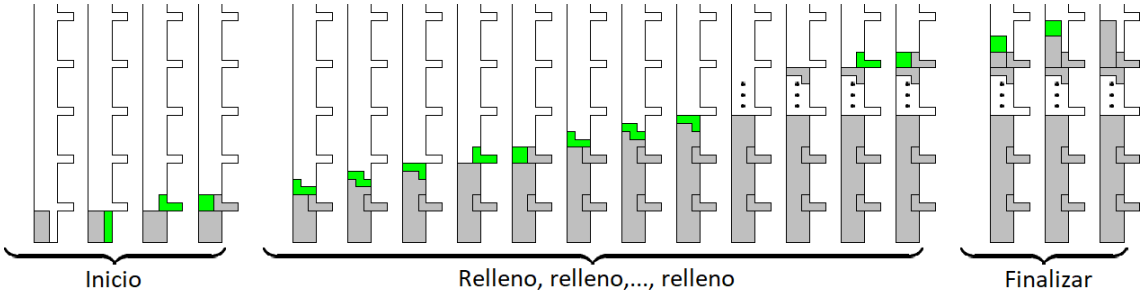


Figura 2.87: Secuencia de piezas de Tetris para un elemento a_i con su forma esperada de acomodo para la reducción de del problema 3-partición.

Por otro lado, la secuencia de control aparece una vez que hayan terminado las piezas correspondientes a los elementos de U, y consiste en las siguientes piezas:

- s piezas $\{l\}$ consecutivas.
- 1 pieza $\{Ld\}$.
- $(3 \cdot T)/2 + 5$ piezas $\{l\}$ consecutivas (como T es par, el resultado es un número natural).

En la figura 2.88 se observa la forma en que se espera que secuencia de piezas de control se acomoden en el escenario.

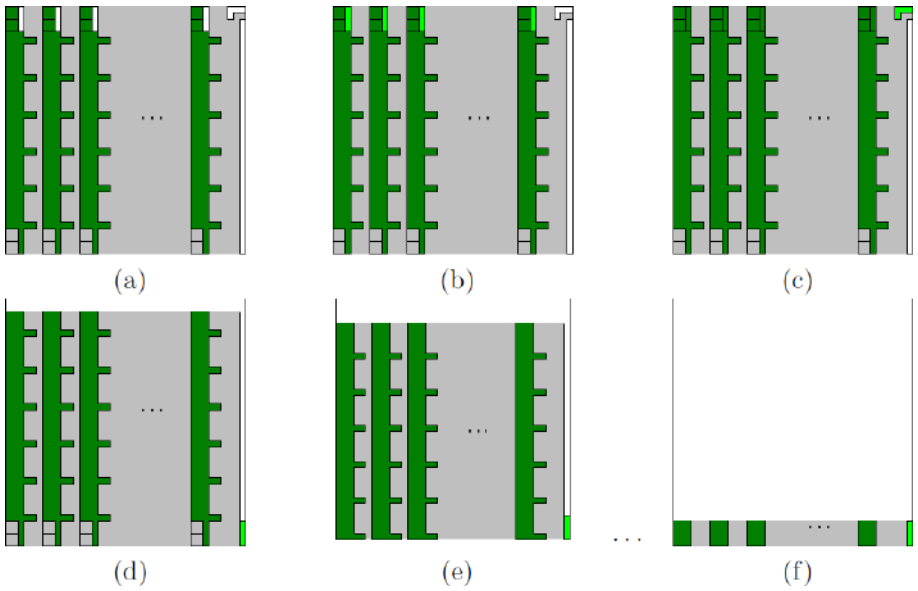


Figura 2.88: Secuencia de piezas de control de Tetris con su forma esperada de acomodo para la reducción de del problema 3-partición.

Una vez explicada la estructura de la reducción, hace falta demostrar que esta reducción es correcta. Esto es, el conjunto U admite una partición en s subconjuntos de tres elementos cuya suma sea T si y sólo si, el escenario mostrado de Tetris puede limpiarse con la secuencia de piezas dada.

Primero se supondrá que el conjunto U admite una partición en s subconjuntos de tres elementos cuya suma sea T.

En este caso, como se explicó anteriormente, a cada uno de los s conjuntos le corresponden seis columnas consecutivas, iniciando con la primera columna. En cada espacio deben acomodarse las piezas correspondientes a tres elementos.

En la figura 2.88 se muestra la forma en que pueden acomodarse estas piezas en cada espacio. Como se observa, cada secuencia de piezas del elemento a_i aumenta la altura del fondo de la cubeta en $6 \cdot a_i + 6$. Si por cada espacio se acomodan las piezas de cada subconjunto $A_j = \{a_i, a_j, a_k\}$, la pieza más baja de ese espacio quedará a altura $6 \cdot T + 18 + 4$, pues $a_i + a_j + a_k = T$.

Esto hará que cuando todas las piezas de los elementos de U hayan sido acomodadas, en cada espacio se tenga la cubeta hasta arriba, con dos espacios vacíos hasta arriba la tercera columna del espacio. La forma en la

que el escenario se encuentra al término de estos turnos, puede verse en la figura 2.88.

De igual forma, en la figura 2.88 se observa la forma en que pueden acomodarse las piezas de la secuencia de control, una vez que el tablero se encuentra en el estado explicado anteriormente. Cada una de las piezas l se posiciona en cada una de las cubetas, por lo que se dejan a las primeras $6 \cdot s$ columnas llenas hasta la fila $6 \cdot T + 22$. La siguiente pieza Ld permite destruir las filas de hasta arriba.

Esto deja el escenario lleno completamente por debajo de la fila $6 \cdot T + 20$, con excepción de la última columna, que quedará completamente vacía. Las siguientes $(3 \cdot T)/2 + 5$ piezas l se apilan en esta última fila, lo que crea una altura de $6 \cdot T + 20$. Esto limpia el escenario por completo.

Por lo tanto, si el conjunto U admite una partición en s subconjuntos de tres elementos cuya suma sea T, el escenario de tetris puede limpiarse con la secuencia de piezas dada.

Para el otro lado de la demostración, se supondrá que el escenario dado puede limpiarse con la secuencia de piezas dada.

Primero hay que notar que, en total el número de cuadros vacíos del escenario por debajo de la fila $6 \cdot T + 22$ es el mismo que el número de cuadros ocupado por las piezas de la secuencia.

Cada espacio correspondiente a los s conjuntos contiene $20 \cdot T + 64$ espacios vacíos. En la primera y segunda columna hay $6 \cdot T + 18$ en cada una, en la tercera hay $6 \cdot T + 22$, mientras que en la cuarta y quinta hay $T + 4$ en cada una. Entonces, sumando todas las columnas hay $6 \cdot T + 18 + 6 \cdot T + 18 + 6 \cdot T + 22 + T + 3 + T + 3 = 20 \cdot T + 64$ cuadros vacíos en estas secciones. Por otro lado, en la sección de control hay $6 \cdot T + 24$ cuadros vacíos. Esto significa que hay $20 \cdot s \cdot T + 64 \cdot s + 6 \cdot T + 24$ filas vacías por debajo de la fila $6 \cdot T + 22$.

Por el lado de las piezas, cada pieza se compone de cuatro cuadros. Cada pieza utiliza 3 piezas para su sección de inicio, $5 \cdot a_i$ piezas para la sección de relleno, y 2 piezas para la sección para finalizar. Sumando todas las piezas de todos los elementos quedan $3 \cdot (3 \cdot s) + 5 \cdot (s \cdot T) + 2 \cdot (3 \cdot s) = 16 \cdot s + 5 \cdot s \cdot T$.

Entonces en total, estas piezas utilizan $4 \cdot (15 \cdot s + 5 \cdot s \cdot T) = 60 \cdot s + 20 \cdot s \cdot T$.

De la sección de control se tienen $s + 1 + (3 \cdot T)/2 + 5$ piezas. Estas piezas utilizan $4(s+1+3 \cdot T+5) = 4 \cdot s+24+6 \cdot T$ cuadros. Por lo tanto, en total la piezas utilizan $60 \cdot s+20 \cdot s \cdot T+4 \cdot s+24+6 \cdot T = 20 \cdot s \cdot T+64 \cdot s+6 \cdot T+24$ cuadros.

Dado que el número de cuadros vacíos por debajo de la fila $6 \cdot T + 22$ es igual al número de cuadros que ocupan las piezas, la única forma de limpiar la pantalla es acomodar las piezas en los espacios.

Para poder llenar por completo los espacios vacíos, es necesario que no se dejen espacios vacíos inalcanzables. Estos espacios son aquellos que no pueden ser llenados por una pieza de la secuencia.

En[5] se detallan todas las formas en que pueden quedar espacios vacíos inalcanzables. Sin embargo, la cantidad de opciones posibles de estas configuraciones de escenario es muy extensa, y se alejan del objetivo de este trabajo. En la figura 2.89 se observan algunas configuraciones con espacios vacíos.

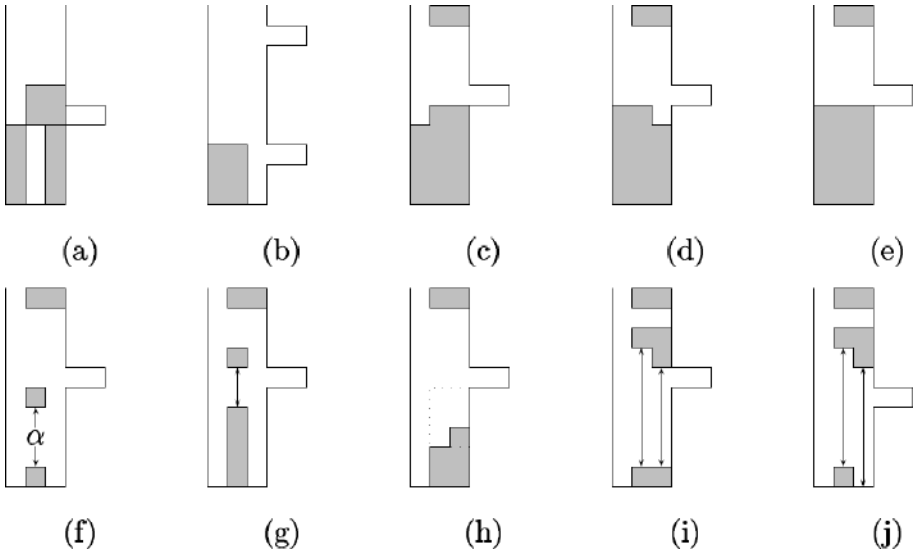


Figura 2.89: Distintas configuraciones de un tablero de Tetris con espacios vacíos inalcanzables en la reducción de del problema 3-partición.

Finalmente, basta con observar que la estrategia de acomodo presentada anteriormente es básicamente la única válida que no genera espacios inalcanzables en algún momento de la ejecución, con algunas variaciones menores permitidas. En[5] se muestra la demostración completa de este

hecho. Sin embargo, esta demostración toma en cuenta una cantidad muy grande de casos que se aleja del objetivo de este trabajo.

Por lo anterior dicho, si existe una forma de limpiar la pantalla, quiere decir que pueden acomodarse todas las piezas correspondientes a tres elementos de U en cada espacio. Tomado estos tres elementos, pueden construirse los subconjuntos $A_1, ..., A_s$ para a 3-Partición.

Por lo tanto, la reducción es válida. A partir de este resultado se puede concluir que decidir si es posible limpiar la pantalla dada una configuración inicial de escenario de Tetris generalizado y una secuencia de piezas, siguiendo sus reglas es *NP-Duro*.

Adicionalmente, en [5] se muestra un modelo matemático más formal para el videojuego de Tetris. Con este modelo, puede afirmarse con más certeza que la reducción mostrada puede realizarse en tiempo y espacio polinomial. Además, este modelo permite verificar que el problema se encuentra en *NP*.

Por lo tanto se puede concluir que decidir si es posible limpiar la pantalla dada una configuración inicial de escenario de Tetris generalizado y una secuencia de piezas, siguiendo sus reglas es *NP-Completo*.

2.3.3. PSPACE-Dureza y Completitud con técnica

2.3.3.1. Super Mario Bros. (TQBF)

En este caso el problema de decisión a resolver es decidir si es posible terminar un nivel dado de Super Mario Bros. generalizado siguiendo sus reglas.

Para probar la *PSPACE-Completitud* del problema, es necesario probar su *PSPACE-Dureza* y pertenencia en *PSPACE*.

Como se menciona en [9] y en [1], el problema de decidir si es posible terminar un nivel dado de Super Mario Bros. generalizado siguiendo sus reglas se encuentra en *PSPACE*, pues el videojuego cumple las características requeridas en la técnica de pertenencia en *PSPACE* explicada anteriormente.

En[9] se muestra la siguiente demostración de Super Mario Bros. generalizado donde se utilizará la técnica para *PSPACE-Dureza* de reducción de *TQBF*. El componente de inicio no es necesario. El componente de final sólo necesita contener una meta. Se requiere que Mario se encuentre siempre en estado pequeño, por lo que no se encontrarán super champiñones en la construcción y se supondrá que Mario inicia en este estado.

Si bien en la demostración de *NP-Dureza* utilizando la técnica de reducción de *3-SAT* se presenta un componente de cruce, este depende de que Mario se mantenga en estado grande durante la construcción. En esta demostración se utiliza un componente de cruce en el que se requiere que Mario se encuentre en estado pequeño.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de puerta y componente de cruce. Estos componentes se presentan en[9].

Componente de cruce:

El componente de cruce se observa en la figura 2.90. Este componente cuenta con dos entradas y dos salidas. Las entradas se encuentran en la parte superior y las salidas en la parte inferior. Si Mario entra por la entrada izquierda, puede llegar a la salida derecha avanzando hacia ella mientras cae. Análogamente al entrar por la entrada derecha, Mario puede llegar a la salida izquierda.

Si Mario intenta llegar a la salida izquierda desde la entrada izquierda, no podrá llegar pues requiere detenerse y dar la vuelta en el aire, con lo que no tendrá suficiente velocidad para llegar a la salida, como se muestra en la figura 2.90b. Lo mismo pasará si Mario intenta llegar a la salida derecha desde la entrada derecha.

Al caer, Mario quedará atrapado en el fondo del componente, pues es lo suficientemente profundo para que Mario no pueda salir saltando, y no podrá seguir avanzando.

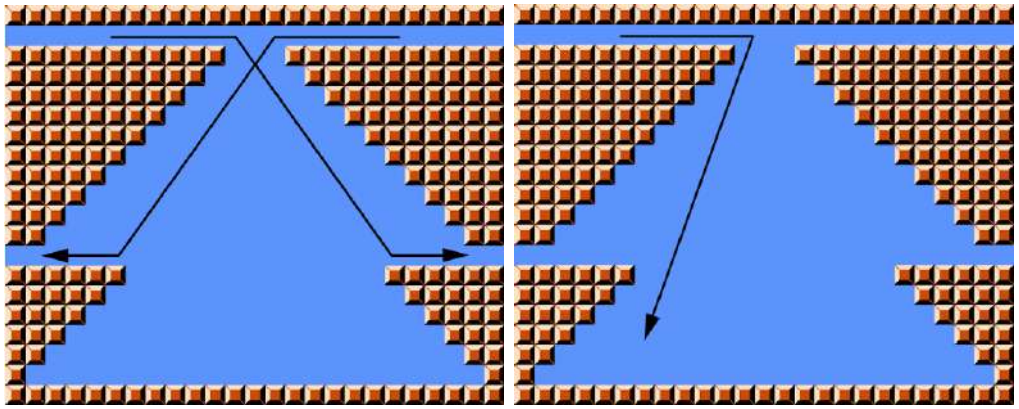


Figura 2.90: Componente de cruce de Super Mario Bros. para la estructura de *TQBF*.

Componente de puerta:

El componente de puerta se observa en la figura 2.91. El spiny que se muestra camina a través del camino de cuatro cuadros de izquierda a derecha y rebota para regresar y continuar caminando en esos cuatro cuadros. Las flamas que se observan representan barras de fuego, se presenta solamente la primera flama de la barra, que sirve como pivote para la misma.

Estas barras de fuego están sincronizadas de tal manera que Mario puede esquivarlas para atravesar cualquier camino. La función de estas barras es evitar que Mario pase del camino para cerrar el camino para transitar y viceversa. Además, sirven para que el componente sea resistente a algunos glitches que existen en el videojuego original.

Cuando el spiny se encuentre por encima de un bloque de ladrillos, Mario puede saltar por debajo y lanzarlo al otro lado del componente. En la figura 2.91 se muestra el componente en estado abierto. En este estado la única forma de transitar por el camino para cerrar es lanzar al spiny a otro lado del componente, lo cual cambia el estado a cerrado. Como se puede observar, en estado abierto sólo las barras de fuego presentan obstáculo para que Mario atraviese el camino para transitar.

El estado cerrado se presenta cuando el spiny se encuentra en el camino para transitar. En este estado el camino para cerrar puede transitarse sin problema. El camino para abrir permite lanzar al spiny al otro lado del

componente y dejarlo en estado cerrado. Por otro lado, en estado cerrado el spiny bloquea el camino para transitar y Mario no puede atravesarlo sin ser golpeado.

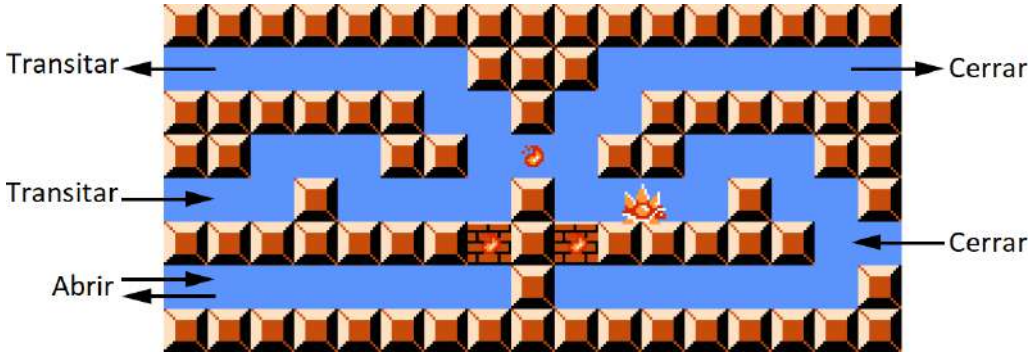


Figura 2.91: Componente de puerta de Super Mario Bros. para la estructura de *TQBF*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si es posible terminar un nivel de Super Mario Bros. generalizado siguiendo sus reglas es *PSPACE-Completo*.

2.3.3.2. Donkey Kong Country (TQBF)

En este caso el problema de decisión a resolver es decidir si es posible terminar un nivel dado de Donkey Kong Country generalizado siguiendo sus reglas.

Para probar la *PSPACE-Completitud* del problema, es necesario probar su *PSPACE-Dureza* y pertenencia en *PSPACE*.

Como se menciona en[1], el problema de decidir si es posible terminar un nivel dado de Donkey Kong Country generalizado siguiendo sus reglas se encuentra en *PSPACE*, pues el videojuego cumple las características requeridas en la técnica de pertenencia en *PSPACE* explicada anteriormente.

En[1] se muestra la siguiente demostración de Donkey Kong Country generalizado donde se utilizará la técnica para *PSPACE-Dureza* de reducción de *TQBF*. El componente de final sólo contiene la meta del nivel. El componente de cruce y de inicio son los mismos que los utilizados en la en

la demostración para *NP-Dureza* de este mismo videojuego utilizando la técnica de reducción de *3-SAT*.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de puerta. Este componente se presenta en [1].

Componente de puerta:

El componente de puerta se observa en la figura 2.92. Los zingers que están agrupados y marcados con rojo, corresponden a zingers con un patrón de movimiento de izquierda a derecha. Todos estos zingers se mueven a la misma velocidad, la cual es una que le permite a Donkey Kong realizar las acciones que se explican posteriormente. Los zingers sin marca son zingers estáticos. Las flechas rojas punteadas indican la trayectoria de los dos grupos de zingers con patrón de movimiento. El suelo del escenario está congelado.

En la figura 2.92, el componente se encuentra en estado cerrado. En este estado, si Donkey Kong llega por la entrada del camino para transitar, será lanzado por el barril cañon y quedará rebotando encima de la llanta. Como no hay espacio para moverse a la izquierda o derecha, Donkey Kong no podrá avanzar a la salida del camino para transitar.

Si Donkey Kong llega por la entrada del camino para cerrar con la puerta en estado cerrado, podrá subir por la cuerda y llegar a la salida sin problema cuando los zinger estén por completo a la derecha.

Para cambiar a estado abierto, Donkey Kong debe llegar desde el camino para abrir, moverse a la izquierda cuando los zingers se lo permitan, y mover la llanta cuesta arriba. El movimiento de los zingers le permite subir la llanta por toda la plataforma inclinada hacia arriba.

Una vez abierta, si Donkey Kong llega por la entrada del camino para transitar, podrá moverse a la salida de este camino cuando el movimiento de los zinger se lo permitan. Si Donkey Kong intenta ir a la entrada o salida del camino para cerrar, los zingers marcados con rojo se lo impedirán.

Si Donkey Kong llega desde la entrada del camino para cerrar, se verá obligado a empujar la llanta para llegar a la cuerda que lo lleva a la salida. Esto obliga a Donkey Kong a cambiar el estado a cerrado para avanzar por el camino para cerrar.

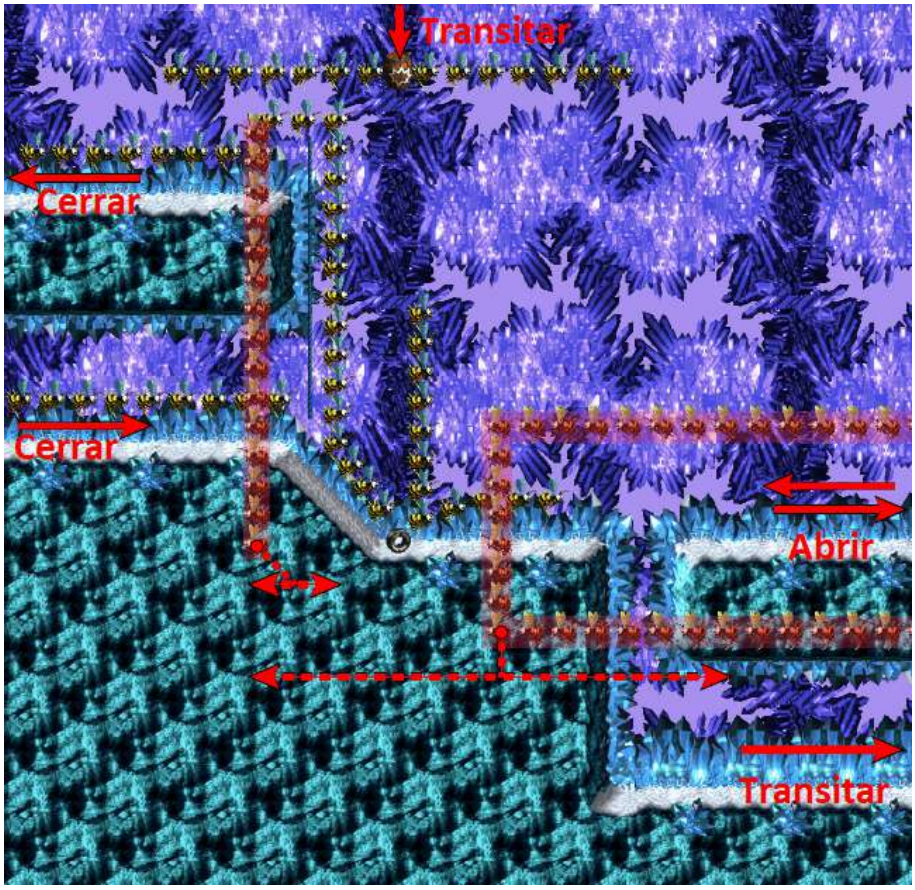


Figura 2.92: Componente de puerta de Donkey Kong Country para la estructura de *TQBF*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si es posible terminar un nivel de Donkey Kong Country generalizado siguiendo sus reglas es *PSPACE-Completo*.

Adicionalmente, en[1] se presentan demostraciones con esta técnica para algunas secuelas de Donkey Kong Country. Sin embargo, estas demostraciones dependen de algunas reglas introducidas en las mismas secuelas.

2.3.3.3. The Legend of Zelda: A Link to the Past (TQBF)

En este caso el problema de decisión a resolver es decidir si es posible terminar un nivel dado de The Legend of Zelda: A Link to the Past generalizado siguiendo sus reglas.

Para probar la *PSPACE-Completitud* del problema, es necesario probar su *PSPACE-Dureza* y pertenencia en *PSPACE*.

Como se menciona en[1], The Legend of Zelda: A Link to the Past se encuentra en *PSPACE*, pues el videojuego cumple las características requeridas en la técnica de pertenencia en *PSPACE* explicada anteriormente.

En[1] se muestra la siguiente demostración de The Legend of Zelda: A Link to the Past donde se utilizará la técnica para *PSPACE-Dureza* de reducción de *TQBF*. El componente de final sólo contiene la meta del nivel. El componente de cruce es el mismo que el utilizado en la en la demostración para *NP-Dureza* de este mismo videojuego utilizando la técnica de reducción de *3-SAT*.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de inicio y componente de puerta. Estos componentes se presentan en[1].

Componente de inicio:

En este caso, se muestra el componente de inicio junto con el componente de puerta en la figura 2.93. Esto es porque el componente de inicio influye solamente en los componentes de puerta. Para la construcción se necesita que las puertas del videojuego marcadas con un número 2 que se muestran en el componente de puerta inicien abiertas. Como se explicó en las reglas, las puertas del videojuego siempre inician cerradas.

Para abrir sólo las puertas necesarias, Link atravesará primero todos los caminos de inicialización que se muestra en el componente de puerta. El interruptor y el teletransportador se encuentran en una plataforma de nivel inferior. Para que Link pueda atravesar el camino de inicializar, Link necesita caer a la plataforma de nivel inferior. Al caer a esta plataforma, Link activará el interruptor que abre las puertas requeridas.

Una vez presionado este interruptor, el único movimiento que se puede realizar es moverse al teletransportador. al moverse al teletransportador, Link no podrá volver a presionar el interruptor, por lo que la puerta quedará abierta. Por el estado en el que inician los pilares, una vez que Link sale del teletransportador, solo podrá avanzar hacia el camino de inicialización del siguiente componente de puerta.

El camino de inicialización del último componente de puerta está conectado con la porción de nivel que se muestra en la parte superior de la figura 2.93. En esta sección se encuentra un cristal que cambia el estado de los pilares. Para poder avanzar se requiere activar este cristal. Al activar el cristal, se cierra el camino por el que Link llegó y cambia los pilares de los componentes de puerta, cerrando el acceso al camino de inicialización de todas las puertas.

Posteriormente se encuentra un teletransportador, que conecta con el inicio de la construcción. Al activar este teletransportador, se pierde completamente el acceso al componente de inicio.

Componente de puerta:

Como se mencionó con el componente anterior, el componente de puerta se muestra en la figura 2.93. Al pasar en cualquier momento por el componente de puerta, los pilares se encuentran en el estado contrario al mostrado en la figura 2.93. El estado cerrado corresponde a cuando las puertas del videojuego marcadas con el número 1 se encuentran cerradas y las puertas del videojuego marcadas con el número 2 se encuentren abiertas. Por el componente de inicio, todos los componentes de puerta se encuentran en estado cerrado en un principio.

Si el componente se encuentra en estado cerrado, el camino para transitar no se puede atravesar, pues hay una puerta del videojuego cerrada enmedio del camino. En este estado, si Link llega por la entrada del camino para cerrar, la única opción para avanzar es atravesar la puerta cercana marcada con el número 2, pues la otra está cerrada. Al atravesar esta puerta se debe pasar por el teletransportador para avanzar.

Al salir del teletransportador, como el camino de inicialización está cerrado, al igual que la puerta del videojuego marcada con el número 2, la única opción es ir a la salida del camino para cerrar.

Al llegar por la entrada del camino para abrir en estado cerrado, la única opción para avanzar es atravesar la puerta cercana marcada con el número 2, pues la otra está cerrada. Al entrar por esta puerta, para avanzar será necesario caer a la plataforma de nivel inferior que contiene el interruptor que abre las puertas marcadas con 1. Al caer se activará el botón y abrirá las puertas marcadas con 1.

Una vez activado el botón, será necesario caminar por el teletransportador que se encuentra al lado para poder avanzar. Al hacerlo no se podrá volver a activar el botón.

Lo siguiente que se tiene que hacer es activar el botón que abre las puertas marcadas con 2 y salir por el teletransportador, de forma análoga a como se hizo con el interruptor anterior. Al llegar hasta este punto, el estado del componente habrá cambiado a abierto, pues las puertas del videojuego marcadas con 1 estarán cerradas y las puertas del videojuego marcadas con 2 abiertas.

Por el estado de las puertas del videojuego, sólo se podrá avanzar por la salida del camino para abrir. Por la posición de los pilares, no se puede pasar al camino de inicialización en ningún momento del recorrido.

Si el componente se encuentra en estado abierto, la puerta del videojuego que se encuentra enmedio del camino para transitar estará abierta, y será posible atravesarlo. Si Link llega por la entrada del camino para abrir con el componente de puerta en estado abierto, se realiza un recorrido análogo al realizado al llegar por la entrada del camino para cerrar con la puerta en estado cerrado.

De igual forma, si Link llega por la entrada del camino para cerrar con el componente de puerta en estado abierto, se realiza un recorrido análogo al realizado al llegar por la entrada del camino para abierto con la puerta en estado cerrado.

De esta forma, ambos caminos para abrir y para cerrar, obligan a cambiar de estado al componente de puerta para poder atravesarlos.

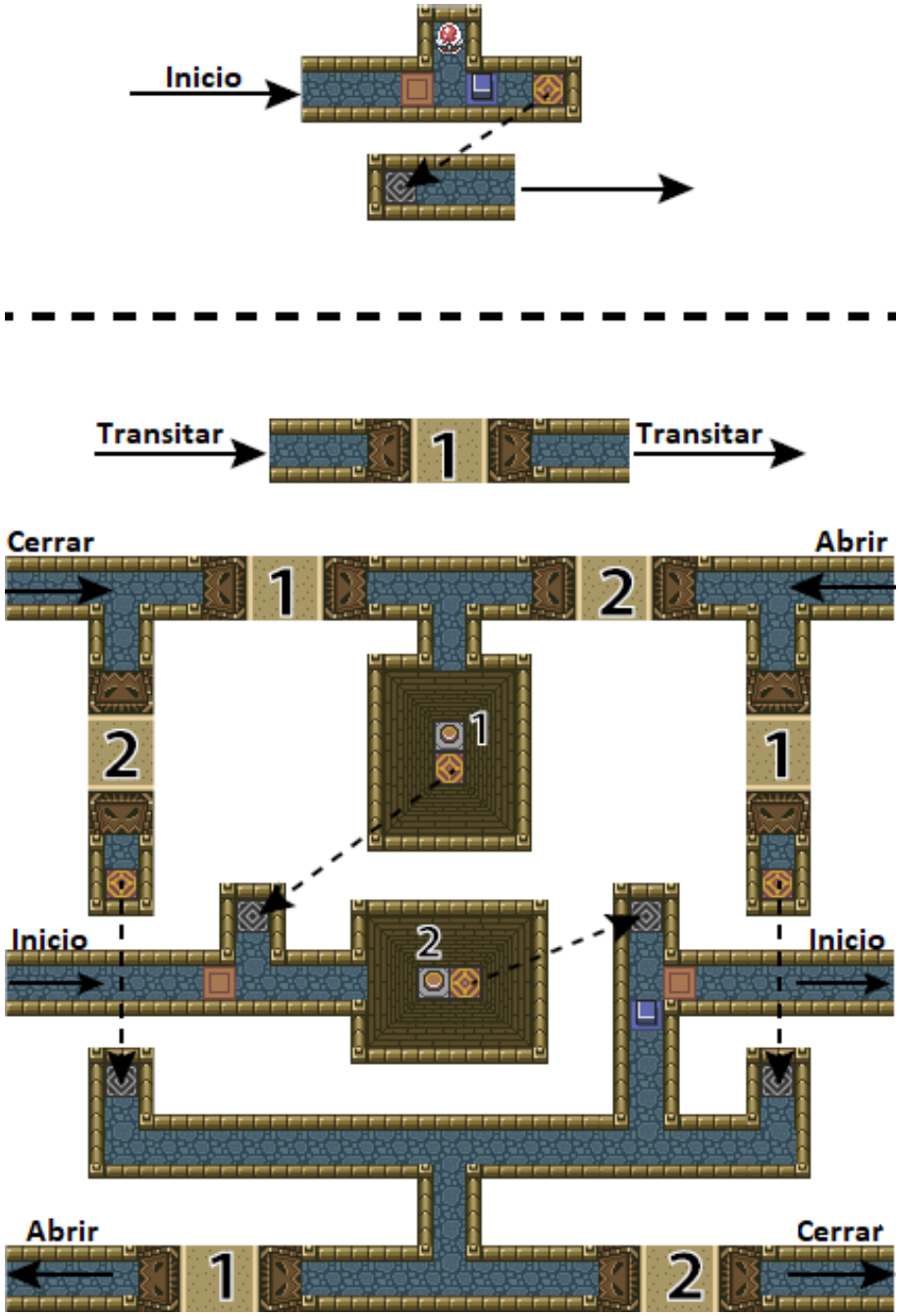


Figura 2.93: Componentes de puerta y de inicio de The Legend of Zelda: A Link to the Past para la estructura de *TQBF*.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si es posible terminar un nivel dado de The Legend of Zelda: A Link to the Past generalizado siguiendo sus reglas es *PSPACE-Completo*.

Adicionalmente, en[1] se menciona que algunos otros títulos de la saga que contienen los mismos elementos pueden utilizar la misma demostración.

2.3.3.4. Portal (Lógica de restricción no determinista)

En este caso el problema de decisión a resolver es decidir si dado un escenario de Portal generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas.

Para probar la *PSPACE-Completitud* del problema, es necesario probar su *PSPACE-Dureza* y pertenencia en *PSPACE*.

En[8], se menciona que el problema de decidir si dado un escenario de Portal generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas se encuentra en *PSPACE*, pues el videojuego cumple las características requeridas en la técnica de pertenencia en *PSPACE* explicada anteriormente.

En general, se explica con mayor detalle en[8] las razones de por qué cada elemento y estado del mismo puede ser representado en espacio polinomial.

En[8] se muestra la siguiente demostración de Portal donde se utilizará la técnica para *PSPACE-Dureza* de reducción de lógica de restricción no determinista. Dado que en Portal pueden crearse pasillos como los necesarios para aplicar esta técnica, basta con explicar la implementación de interruptores que controlan al menos siete puertas, y las puertas que son controladas por un interruptor.

Para implementar los interruptores, se utiliza un cuarto con dos botones sensibles al peso y un bloque. Uno de los botones controla a las puertas que inician cerradas que corresponden a la arista del interruptor, y el otro botón controla a las puertas que inician abiertas que corresponden a la arista del interruptor.

El bloque en un inicio se encuentra sobre el botón que controla a las puertas que inician abiertas. De esta manera, para cambiar el estado del interruptor basta con cambiar el bloque de botón. Para obligar a que al salir del cuarto del interruptor se encuentre en un estado válido, se agrega una puerta adicional en la entrada del cuarto interruptor.

Esta puerta se abre al activar al menos uno de los botones sensibles al peso. Esta construcción también impide que los bloques salgan del cuarto interruptor. Esta implementación de interruptor se observa en la figura 2.94.

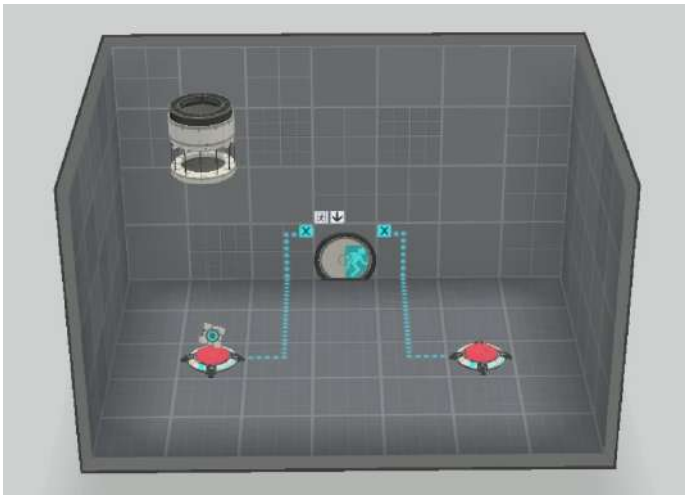


Figura 2.94: Implementación de interruptor de Portal para la estructura de reducción de lógica de restricción no determinista.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un escenario de Portal generalizado, es posible llegar desde un punto inicial a un punto final del mismo, siguiendo sus reglas es *PSPACE-Completo*.

Adicionalmente, en[8] se mencionan otras formas de aplicar la técnica de reducción de lógica de restricción para probar la *PSPACE-Dureza* de Portal. Además, se menciona que la secuela Portal 2 cuenta con todos los elementos necesarios para realizar las mismas demostraciones.

2.3.4. PSPACE-Dureza y Completitud sin técnica

2.3.4.1. Mario Kart 64

En este caso se utilizará el modo de versus con dos jugadores, y el problema de decisión a resolver es decidir si dado un escenario de Mario Kart 64 generalizado en modo versus con dos jugadores existe una estrategia que

fuerce una victoria al primer jugador, con ambos jugadores siguiendo sus reglas.

Para esta demostración de Mario Kart 64 presentada en[4], se reducirá el problema *TQBF* con cuantificadores intercalados iniciando con un cuantificador existencial y con tres literales en cada cláusula. Este problema es mencionado en una de las técnicas utilizadas en este trabajo para la demostración de problemas *PSPACE-Duros*, pero en este caso no se utilizará esa técnica.

En general, se usará una estructura similar al utilizado en la técnica para *NP-Dureza* de reducción de *3-SAT*. Sin embargo, el jugador 1 tendrá acceso a las variables cuantificadas existencialmente y el jugador 2 tendrá acceso a las variables cuantificadas universalmente. Se nombrará a las variables existenciales como $x_1, ..x_i, ..x_{n/2}$ y a las variables universales como $y_1, ...y_i, ...y_{n/2}$.

En la figura 2.95 se observa un diagrama con la estructura de la construcción. Este diagrama se muestra simplificado en la parte de acceso de las variables a las cláusulas, pues esto se explica con más detalle en la técnica para *NP-Dureza* de reducción de *3-SAT*. Como puede observarse, el jugador 1 tiene un camino alternativo, llamado camino de acuerdo.

Este camino tiene el propósito de que el jugador 1 lo utilice si es que el jugador 2 no sigue el camino explicado. Este camino tiene un tiempo de recorrido óptimo más grande que el camino que toma el jugador 2, pero menor al tiempo que tomaría al jugador 2 terminar su recorrido si se detiene o avanza en reversa en algún momento.

El otro camino, el cual contiene la sección de revisión de las cláusulas, tiene un tiempo de recorrido óptimo menor al del camino del jugador 2. Sin embargo, si el jugador 1 golpea al menos un plátano, su tiempo de recorrido será mayor al óptimo del jugador 2.

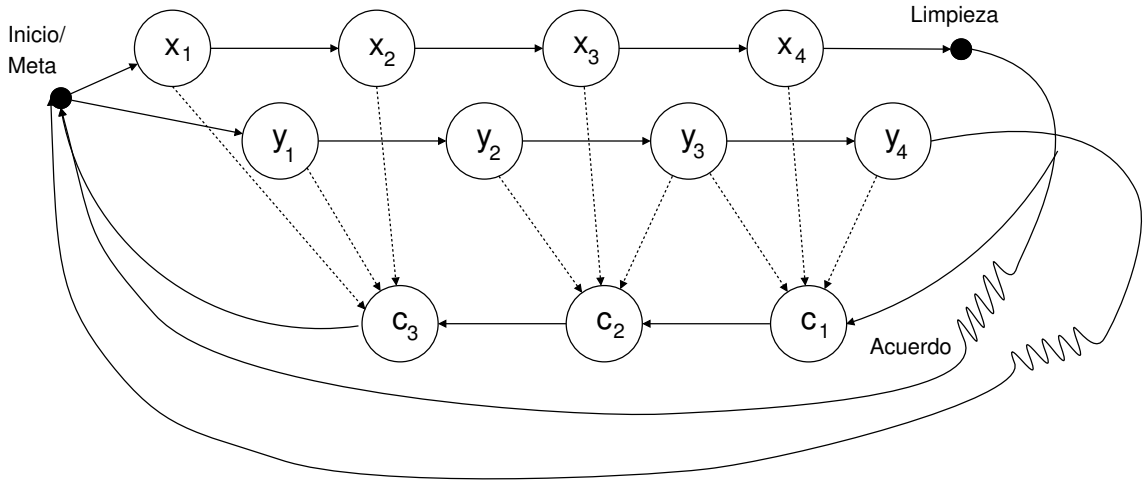


Figura 2.95: Estructura de la reducción de *PSPACE-Dureza* de Mario Kart con dos jugadores.

La idea general de esta demostración es que ambos jugadores tengan las opciones de asignar valores a sus variables correspondientes. Para ponerlo simple, esto se traduce a que siempre exista una decisión ganadora del jugador uno, para todas las decisiones que tome el jugador 2. De esta forma, si el jugador asignado a las variables existenciales siempre puede encontrar un camino de decisiones sin importar cuáles decisiones tome el jugador asignado a las variables universales, la fórmula ϕ es verdadera. Esta idea se explica mas detalladamente en[2], definido como el juego *QBF*.

Dada la construcción, se necesitan los mismos componentes que en la estructura de la técnica para *NP-Dureza* de reducción de *3-SAT*, con la diferencia que se necesitan dos componentes de variable distintos. El componente de inicio y final es el mismo, pues es una pista cíclica. En este caso, es necesario que la meta e inicio sea muy larga, de tal manera que pueda extenderse a dos caminos disjuntos. Se necesita que cada jugador inicie en un camino que le permita recorrer sus propias variables y que no les permita entrar al camino del otro.

A diferencia de la otra demostración, en este caso es necesario que puedan existir plátanos en el suelo del escenario al inicio, o en todo caso, agregar un camino de inicialización donde se agreguen estos plátanos al escenario. De cualquier forma, en esta demostración se supondrá que los plátanos mostradas en los componentes se encuentran ahí al iniciar la carrera. También se requiere que ambos jugadores inicien la carrera al mismo tiempo.

Para el componente de cruce se usará el mismo que el mostrado en la demostración para *NP-Dureza* de este mismo videojuego utilizando la técnica de reducción de *3-SAT*.

Una vez explicada la estructura de la reducción, basta con explicar la construcción de los componentes de variable y el componente de cláusula. Estos componentes se presentan en[4].

Componentes de variable:

El componente de variable existencial será el mismo que el componente de variable el utilizado en la demostración para *NP-Dureza* de este mismo videojuego utilizando la técnica de reducción de *3-SAT*.

El componente de variable universal será parecido al componente de variable existencial. La única diferencia es que en las cajas de los caminos conectados a la sección de activación de los componentes de cláusulas tienen plátanos en lugar de caparazones verdes. En la figura 2.96 se muestra el componente de variable universal.

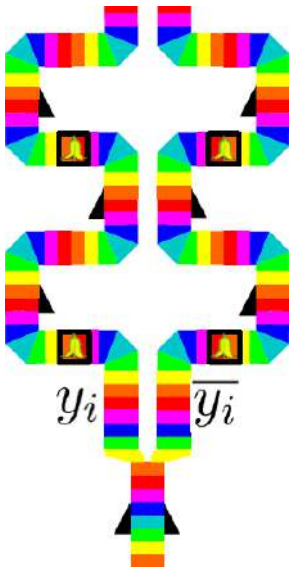


Figura 2.96: Componente de variable universal de Mario Kart 64 para la demostración de *PSPACE-Dureza*.

Finalmente, para que ambos jugadores puedan tomar sus decisiones tomando en cuenta las decisiones del otro, es necesario que puedan observar las decisiones de su contrincante. Por esta razón, cuando un jugador se encuen-

tra atravesando un componente de variable, el otro jugador se encuentra en una zona de observación en una plataforma superior.

Esta zona se encuentra rodeada por una barda para impedir que se lancen objetos o se intente pasar a la otra plataforma. La zona de observación se muestra en la figura 2.97. La parte oscura corresponde a una plataforma a una altura menor.

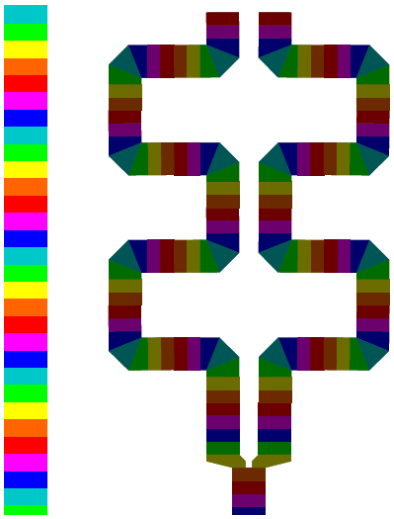


Figura 2.97: Zona de observación de los componentes de variable de Mario Kart 64 para la demostración de *PSPACE-Dureza*.

Componente de cláusula:

Al igual que en la demostración para *NP-Dureza* de este mismo videojuego, se utilizará el componente de limpieza al inicio del camino de revisión. Tanto este componente de limpieza, como el camino de revisión, sólo serán recorridos por el primer jugador. Por otro lado, el jugador 2 recorrerá un camino vacío y sin bifurcaciones.

El componente de cláusula también es parecido al utilizado en la demostración para *NP-Dureza* de este mismo videojuego utilizando la técnica de reducción de *3-SAT*. La diferencia es que en este caso, los caminos de las literales de las variables universales no tienen plátanos en un inicio.

En este caso, el jugador 2 tendrá que lanzar los plátanos desde su camino por la sección de activación del componente. Por otro lado, el jugador 1

podrá destruir los plátanos desde su camino por la sección de activación del componente.

Así, el jugador 1 asignará el valor positivo las literales que elija y el jugador 2 asignará el valor negativo a las que él elija. Si el jugador 1 siempre puede tomar una serie de decisiones que le permitan ganar, entonces la fórmula booleana cuantificada es verdadera, por lo explicado al principio de la demostración. El componente de cláusula se observa en la figura 2.98.

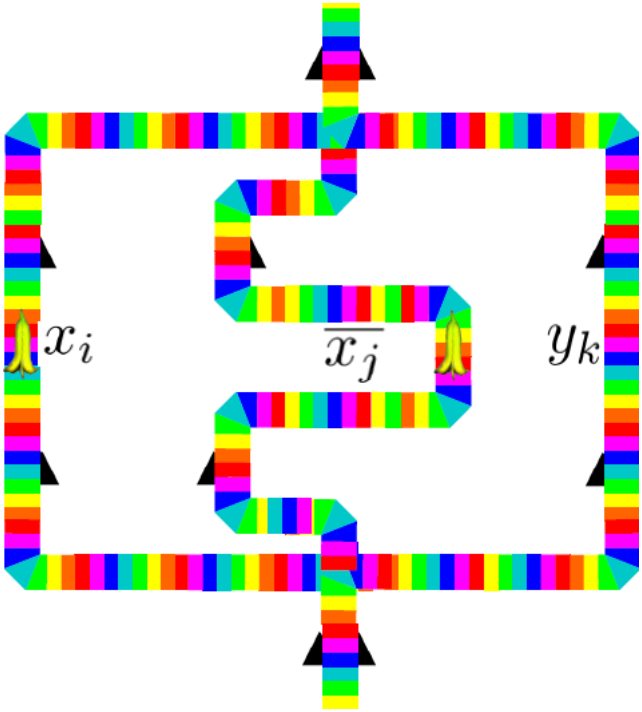


Figura 2.98: Componente de cláusula de Mario Kart 64 para la demostración de *PSPACE-Dureza*.

A partir de esta construcción, se puede concluir que decidir si dado un escenario de Mario Kart 64 generalizado en modo versus con dos jugadores existe una estrategia que fuerce una victoria al primer jugador, con ambos jugadores siguiendo sus reglas es *PSPACE-Completo*.

Al igual que con la demostración de *NP-Dureza*, en [4] se menciona que esta demostración puede ser utilizada para todos los videojuegos de la misma saga que sean 3D. También de la misma forma, no se puede aplicar la demostración en los videojuegos que no son en 3D, pues no cuentan con plataformas a distintas elevaciones.

2.4. Resumen

En este capítulo se presentan problemas de decisión en videojuegos estudiados en trabajos previos que han sido demostrados como *NP-Duros*, *NP-Completos*, *PSPACE-Duros* o *PSPACE-Completos*.

En principio, se explican las reglas generales de cada uno de los videojuegos estudiados, en una versión generalizada que pueda ser estudiada como un problema discreto. En cada uno de ellos se especifican solamente los elementos que influyen en las demostraciones presentadas.

Posteriormente se presentan una serie de técnicas que se utilizan en las demostraciones presentadas. Estos ftécnicas están clasificados dependiendo si están diseñados para demostrar *NP-Dureza*, *PSPACE-Dureza* o pertenencia en *PSPACE*.

Finalmente se presentan las demostraciones de *NP-Dureza* y *PSPACE-Dureza* o *Completitud* presentadas en otros trabajos previos sobre los videojuegos mencionados. Estas demostraciones se clasifican dependiendo de si se utilizó una técnica explicado anteriormente y de si se demuestra su Dureza o Completitud en *NP* o en *PSPACE*.

3 Resultados de dureza originales

En esta sección, se presentarán una variedad de videojuegos estudiados en este trabajo, cuya dureza computacional no ha sido estudiado en trabajos anteriores.

Al igual que en la sección de videojuegos analizados anteriormente, primero se explicarán las reglas de estos videojuegos y posteriormente se presentarán las demostraciones originales de *NP-Dureza* y *PSPACE-Completitud*.

Estos videojuegos son Crash Bandicoot, The Binding of Isaac y Sheep Raider para *NP-Dureza*, y Klonoa: Empire of Dreams para *NP-Dureza* y *PSPACE-Completitud*.

3.1. Reglas de videojuegos estudiados en pruebas originales

En esta sección se presentarán las características, reglas y elementos de los videojuegos con demostraciones originales de este trabajo. Al igual que en la sección de videojuegos previamente estudiados, se presentarán solamente los elementos que influyen en el desarrollo de las demostraciones.

Además, algunos de estos elementos se generalizan para poder tener una versión del videojuego que pueda expresar instancias con dureza computacional. De igual forma que en la sección de videojuegos previamente estudiados, para las versiones generalizadas de estos videojuegos se supondrá que algunos elementos son no acotados. Estos elementos se especificarán en cada videojuego.

3.1.1. Crash Bandicoot

Crash Bandicoot es una serie de videojuegos de plataformas en 3D, originalmente desarrollado para la consola Playstation en 1996. En este videojuego, el personaje principal Crash debe recorrer un nivel desde un punto inicial al final[3]. En la figura 3.1 puede observarse una imagen del videojuego original en ejecución.

Figura 3.1: Crash Bandicoot en ejecución.

Crash puede moverse hacia cualquier dirección en 360 grados a la redonda, con una velocidad determinada. Puede saltar hacia arriba o hacia alguna dirección con una altura y distancias determinadas. Puede realizar un ataque que dura una pequeña fracción de tiempo, en este periodo corto de tiempo, los enemigos y objetos que toque a Crash son destruidos.

En el escenario existen varios elementos que pueden interactuar con Crash, al ser tan variados, se mencionan sólo los necesarios para este trabajo. Los escenarios de esta versión generalizada pueden ser de tamaño no acotado, así como incluir una cantidad no acotada de elementos en el escenario.

En el escenario hay plataformas sobre las que Crash puede moverse, al igual que secciones de vacío en las cuales si se camina sobre ellas, Crash caerá y morirá. Se puede saltar por encima del vacío para evitar que caiga.

Existen distintos tipos de cajas alrededor de los niveles, las utilizadas en este trabajo son caja *TNT*, caja metálica, caja inactiva y caja activadora. Todas las cajas pueden encontrarse a nivel del suelo o por encima de este.

La caja de la figura 3.2 es una caja *TNT*. Estas cajas pueden tocarse por Crash sin problema, pero si Crash las ataca, se genera una explosión y muere inmediatamente. Si se cae de un salto por encima de una de estas cajas, o se salta por debajo de una y se toca desde abajo, empieza una cuenta regresiva de 3 segundos, cuando la cuenta llega a 0, la caja explota.

Si una caja *TNT* explota, los objetos que se encuentran en el rango de su explosión reciben un golpe, en particular si otra *TNT* se encuentra al lado de otra, esta otra explotará y generará otra explosión, lo que puede generar una explosión en cadena. Si la explosión toca a Crash, recibe un golpe. Si la explosión toca un objeto, lo destruye y Crash no podrá tomarlo.

Figura 3.2: Caja *TNT* en Crash Bandicoot.

La caja de la figura 3.3 es una caja de metal. Una caja de metal no puede ser destruida ni movida por ningún ataque, ya sea de Crash o de una explosión. Tampoco causa daño a Crash, pero puede caminar por encima de ellas.

Figura 3.3: Caja de metal en Crash Bandicoot.

La caja de la figura 3.4 es una caja inactiva. Cuando una caja se encuentra en este estado, es intangible y no tiene ningún efecto sobre Crash o el escenario.

Figura 3.4: Caja inactiva de Crash Bandicoot antes y después de activarse.

La caja de la figura 3.5 es una caja activadora. Estas cajas transforman un conjunto dado de cajas inactivas en otras cajas específicas, ya sea de metal, *TNT* u otra activadora. Para activarla se debe saltar por debajo de ella, caer sobre ella o golpearla con un ataque. Una vez activada, esta caja se convierte en una caja de metal.

Figura 3.5: Caja activadora de Crash Bandicoot antes y después de activarse.

En principio, Crash cuenta con un punto de vida, por lo que un sólo golpe te quita una vida y te devuelve al principio del nivel o a un punto de control. Existe un objeto llamado Aku-Aku, el cual puede encontrarse en el escenario o dentro de una caja, que permite tener puntos de vida adicionales.

Este objeto es acumulable, al tener uno o dos te permite tener puntos de vida extra. Al recoger una tercera copia del objeto, Crash se volverá invulnerable durante un periodo de tiempo, con lo que destruirá a todo los enemigos y las cajas *TNT* que toque, entre otros elementos no utilizados en las demostraciones de este trabajo.

Una vez que el tiempo de invencibilidad se termine, Crash volverá al estado en el que tiene dos Aku-Aku. Si se toca un cuarto Aku-Aku cuando se está en estado invencible, el tiempo de invencibilidad se reiniciará. Este objeto puede observarse en la figura 3.6.

Figura 3.6: Distintos estados de Aku-Aku. En la primera imágen se muestra a Aku-Aku como objeto en el escenario. La seguda y tercera imagen muestra a crash habiendo tomado una y dos Aku-Aku respectivamente. La última imagen muestra a Crash con invencibilidad temporal al tomar tres Aku-Aku.

En este videojuego se encuentran varios tipos de niveles, para este trabajo se utilizará un tipo específico de nivel. El tipo de nivel que se utilizará se compone de un pasillo muy largo, rodeado por paredes que sirven de límites de nivel. En estos niveles pueden haber también paredes enmedio del nivel, para evitar que se atraviese de un punto específico a otro directamente. En la figura 3.1 puede observarse un nivel con estas características.

Para mayor información del videojuego y sus reglas, consultar [3].

3.1.2. The Binding of Issac: Rebirth

The Binding of Isaac es un videojuego de exploración de mazmorras en vista aérea desarrollado para distintas plataformas en 2014. En este videojuego, el personaje principal Issac debe desplazarse por una serie de cuartos unidos por puertas para llegar a un cuarto final donde encuentra un jefe a derrotar[23]. En la figura 3.7 puede observarse una imagen del videojuego original en ejecución.

Figura 3.7: The Binding of Isaac: Rebirth en ejecución.

Isaac puede moverse en 8 direcciones, arriba, abajo, izquierda, derecha y diagonales a una velocidad determinada. Además, puede disparar mientras se mueve en cualquiera de las 8 direcciones independientemente del movimiento, a una velocidad y distancia determinadas.

Existen muchos elementos en el videojuego original, pero en este trabajo se restringirán los elementos del videojuego con los que se necesitan para las demostraciones. El número de cuartos de un escenario de esta versión generalizada puede ser no acotado, y sus elementos sólo están acotados por el tamaño del cuarto.

En la figura 3.8 se observan los elementos del escenario a utilizar en este trabajo, mientras que en la figura 3.9 se pueden encontrar los objetos recolectables a utilizar en esta versión generalizada.

Todos los objetos ocupan el mismo espacio en el escenario. Se le llamará al espacio que utilizan un cuadro. El espacio que utiliza Isaac es menor al de un cuadro, aproximadamente una cuarta parte de este.

Figura 3.8: Elementos en los cuartos de The Binding of Isaac.

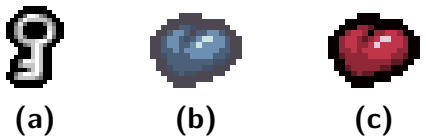


Figura 3.9: Objetos de The Binding of Isaac.

Existen puertas que se encuentran en las orillas de cada cuarto y conectan con otro cuarto.

En cada cuarto puede haber piedras, estas ocupan un cuadro y no pueden destruirse con ataque de Isaac, y tampoco se puede caminar sobre ellas ni disparar a través de ellas. Las piedras pueden verse en la figura 3.8a.

En los cuartos pueden encontrarse bloques metálicos, estos bloques ocupan un cuadro y no pueden ser destruidos. Los bloques metálicos pueden observarse en la figura 3.8c.

En los cuartos también pueden encontrarse púas. Las púas se encuentran en el suelo, se puede caminar sobre ellas, pero al hacerlo causan un golpe de daño a Isaac. Las púas pueden observarse en la figura 3.8d.

Los objetos se pueden encontrar en los cuartos para que Isaac los tome al tocarlos. Estos objetos pueden ser llaves, corazones o corazones de alma.

Las llaves son acumulables y tienen un límite de 99. En esta versión del videojuego generalizada se puede tener una cantidad no acotada de llaves.

Existen puertas cerradas con candados, estas puertas no se abrirán aunque

se destruyan todos los enemigos, para abrirlas se utiliza una llave y se reduce la cantidad de llaves que tienes dependiendo del número de candados de la puerta. Las llaves se observan en la figura 3.9a.

Al inicio Isaac cuenta con tres contenedores de corazón llenos. Cada que Isaac recibe un golpe, pierde la mitad del contenido de un contenedor, y una vez que se queda sin contenido en ningún contenedor, Isaac muere y pierde. Al recibir daño, Isaac tiene un periodo corto de tiempo de invencibilidad.

Dentro de los niveles, pueden encontrarse corazones que recargan el contenido de un contenedor. También pueden encontrarse corazones de alma, los cuales se añaden a la cantidad total de contenedores. Si estos contenedores se vacían desaparecerán, y no pueden rellenarse. Isaac tiene un máximo de 12 contenedores. Si se tiene el máximo, ya no es posible tomar otro corazón de alma.

De igual manera, si se tienen todos los contenedores llenos, no se podrán tomar objetos que los recarguen. Los contenedores de alma se observan en la figura 3.8b, mientras que los corazones se observan en la figura 3.8c.

Cada cuarto tiene un tamaño rectangular. En el videojuego original se tiene una cantidad limitada de tamaños de cuartos, en esta versión generalizada se le permite tener cualquier tamaño rectangular.

En cada cuarto puede haber vacío en el suelo. Isaac no puede caminar sobre el vacío, pero sus ataques pueden pasar por encima del vacío. En la figura 3.10 se observa un escenario con vacío.

Figura 3.10: Vacío en The Binding of Isaac.

En cada cuarto existen distintos enemigos con diferentes patrones de ataque y movimiento. Si un enemigo toca a Isaac le causará daño de un golpe. En general, existen dos tipos de enemigos, los que pueden destruirse y los que

no. En esta versión generalizada sólo se utilizarán enemigos indestructibles llamados wall hugger.

Los wall hugger se mueven infinitamente alrededor de una estructura con una velocidad y dirección fija. Esta estructura puede estar creada por rocas, bloques de metal o incluso los límites del nivel. Los wall hugger utilizan un cuadro de espacio en el escenario.Estos enemigos reinician su posición inicial al entrar al cuarto donde se encuentran. En la figura 3.11 se observan dos wall hugger junto a Isaac.

Figura 3.11: Enemigos wall hugger de The Binding of Isaac.

Existen barriles de *TNT* que ocupan un cuadro. Estos barriles se observan en la figura 3.8b Los barriles de *TNT* pueden ser destruidos por el ataque de Isaac y al destruirse causan una explosión. Las explosiones le causan daño de un golpe a Isaac si se encuentra en el rango de la misma, además de destruir otros barriles de *TNT* y piedras. Las *TNT* pueden observarse en la figura 3.12.

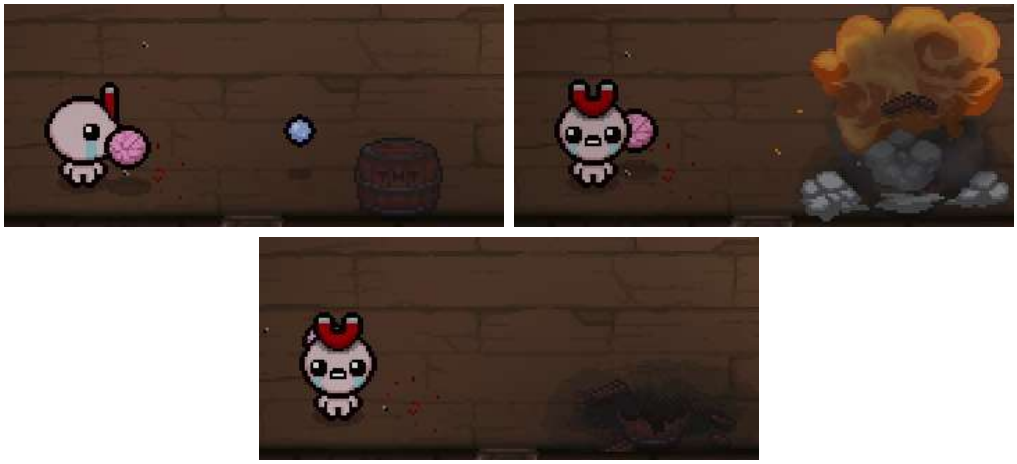


Figura 3.12: Funcionamiento de *TNT* en The Binding of Isaac.

Si una explosión destruye a una piedra por detrás de un cuadro de vacío, este creará un cuadro de piso sobre el que Isaac podrá caminar. Esto se ejemplifica en la figura 3.13.

Figura 3.13: Creación de piso con explosión en The Binding of Isaac.

Una vez que se abandone un cuarto y se regrese a él, los enemigos no destruibles regresarán a su posición inicial, las piedras y barriles de *TNT* destruidos no volverán a aparecer, los objetos que estaban y no fueron tomados seguirán ahí, y los cuadros de piso creados por explosiones seguirán ahí.

Existen habilidades extra permanentes y activables con un botón que le permiten a Isaac romper algunas de estas reglas, para efectos prácticos, estos objetos se considerarán como inexistentes en la versión generalizada del videojuego que se analizará en este trabajo. Existe también una selección de personajes con los cuales se tienen habilidades diferentes, al igual que con los objetos extra, en esta versión del videojuego se supondrá que sólo se puede controlar a Isaac.

Para mayor información del videojuego y sus reglas, consultar[23].

3.1.3. Sheep Raider

Sheep Raider, también conocido como "Sheep, Dog and Wolf", es un videojuego de plataformas en 3D, rompecabezas y sigilo desarrollado originalmente para la consola Playstation y Microsoft Windows en 2001. El objetivo general del videojuego es transportar a una oveja de un punto del nivel a una meta. En este videojuego controlas a un personaje llamado Ralph Wolf, al que se le llamará *el lobo*[17]. En la figura 3.14 se muestra el videojuego en funcionamiento, con el lobo al lado de la meta.

Figura 3.14: Sheep Raider en ejecución.

El lobo puede moverse en cualquier dirección en 360 grados a una velocidad determinada. El lobo puede saltar una distancia y altura determinada, además de poder realizar un doble salto que aumenta esta altura y distancia.

Los escenarios del videojuego pueden estar limitados por paredes o estar limitados por precipicios, sobre los cuales si el lobo trata de caminar, caerá. Los escenarios de esta versión generalizada pueden ser de tamaño no acotado, así como incluir una cantidad no acotada de elementos en el escenario.

En los escenarios, plataformas elevadas a distintas alturas. Dependiendo de la altura de la plataforma, el lobo podrá alcanzarlas con un salto. De la misma manera, existen plataformas inclinadas, por las que el lobo puede caminar para llegar a niveles superiores sin necesidad de saltar. Este tipo de plataformas permite crear caídas largas, al igual que en Portal. En la figura 3.15 se observan plataformas a distintas elevaciones y plataformas inclinadas.

Figura 3.15: Plataformas a distinta elevación y plataformas inclinadas en Sheep Raider.

Dentro de cada nivel puede haber una o más ovejas en alguna parte del nivel, el lobo sólo necesita llevar a una a la meta. El lobo puede cargar a la oveja cuando se encuentra al lado de ella y dejarla enfrente de él en cualquier momento. El lobo puede moverse mientras carga una oveja. Mientras se carga a la oveja, el lobo no puede realizar el doble salto. En la figura 3.16 se observa al lobo con una oveja.

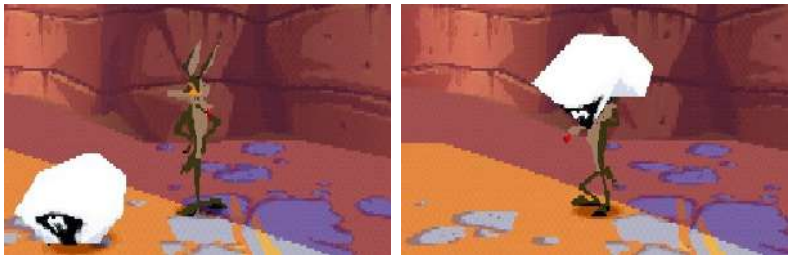


Figura 3.16: Ovejas en Sheep Raider.

En el escenario pueden encontrarse rocas. Estas rocas pueden ser empujadas por el lobo a través de una trayectoria o área definida. El área por la que puede ser empujada una roca, puede incluir ser empujada de una plataforma a una zona de nivel inferior. En la figura 3.17 se observa al lobo con una roca.

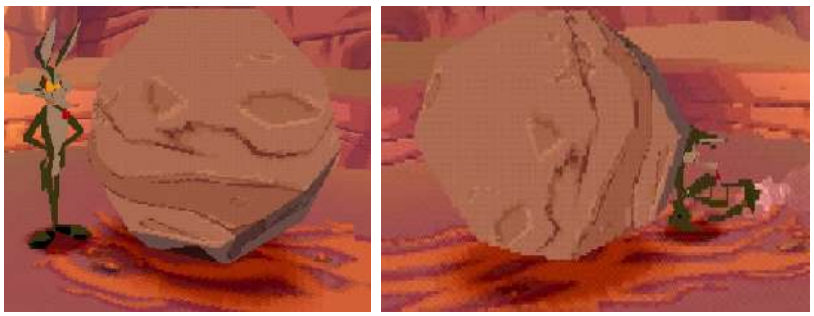


Figura 3.17: Rocas en Sheep Raider.

Existen catapultas en forma de sube y baja. Estas catapultas funcionan poniendo a la oveja del lado bajo y tirando una piedra sobre el lado alto. Al caer, la piedra hará que el lado alto baje y lado bajo suba con fuerza. La fuerza con la que sube la catapulta provoca que la oveja sea lanzada en forma de parábola hacia el lado de donde cayó la piedra.

Si se lanza una oveja de esta manera, aterrizará más allá de la catapulta, por lo que puede utilizarse para subir a la oveja a plataformas elevadas. En la figura 3.18 se observa el funcionamiento de estas catapultas.

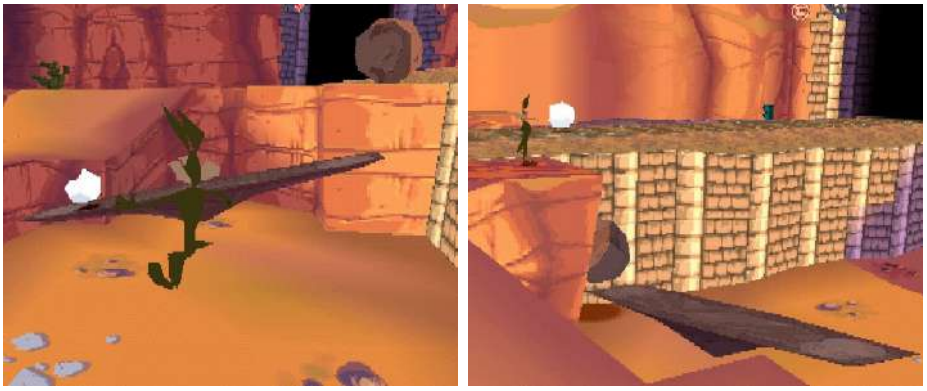


Figura 3.18: Funcionamiento de catapultas en Sheep Raider.

Cuando no hay un objeto en ninguno de los lados de la catapulta, se mantendrá con ambos lados en el aire a una altura media. El salto del lobo es lo suficientemente alto para llegar a la catapulta a media altura, no se necesita un doble salto.

Para mayor información del videojuego y sus reglas, consultar[17].

3.1.4. Klonoa: Empire of Dreams

Klonoa: Empire of Dreams es un videojuego de plataformas en dos dimensiones y rompecabezas desarrollado para la consola game boy advance en 2001. El objetivo general del videojuego es recolectar tres estrellas y llegar a una puerta en una sección del nivel. Para recolectar estas estrellas, basta con tocarlas[16]. En la figura 3.19 se observa el videojuego en funcionamiento, junto con las estrellas objetivo.

Figura 3.19: Klonoa: Empire of Dreams en ejecución.

El personaje principal, Klonoa, puede moverse de izquierda a derecha a una velocidad determinada. Además, puede saltar por una altura y distancia determinada. Klonoa puede flotar por un breve periodo de tiempo, permitiéndole llegar un poco más adelante en sus saltos.

El salto de Klonoa le permite subir a plataformas que miden estrictamente menos que el doble de su altura. La distancia vertical del salto de Klonoa es de tres veces su propia medida y cuatro veces si flota.

Klonoa cuenta al iniciar con tres puntos de vida, el cual es el máximo valor que puede tener. En el escenario hay corazones que te regeneran un punto de vida al tocarlos. En la figura 3.20 se observan estos corazones.

Figura 3.20: Corazones en Klonoa: Empire of Dreams.

Klonoa cuenta con un ataque de corto alcance, que le permite interactuar con algunos elementos del escenario.

Los escenarios son delimitados por paredes, no existen precipicios. Estos escenarios están conformados por varios cuartos. El número de cuartos de un escenario de esta versión generalizada puede ser no acotado, y sus elementos sólo están acotados por el tamaño del cuarto. De la misma forma, el tamaño de un cuarto puede ser no acotado en esta versión generalizada.

Los escenarios pueden estar configurados de tal forma que al entrar a una parte de él, sea imposible escapar de esa zona, en este caso se permite reiniciar el cuarto en el que se encuentre.

En los escenarios pueden encontrarse bloques con cara de enemigo. El bloque puede ser tomado por Klonoa. Cuando Klonoa toma un bloque, se posiciona sobre su cabeza, esto hace que la el espacio que ocupa Klonoa sea del doble de altura que originalmente. Klonoa puede seguir moviéndose a la misma velocidad mientras tiene un bloque sobre su cabeza.

Si Klonoa tiene un bloque sobre su cabeza, no podrá caminar por en medio de ciertas zonas del nivel en las que no cabe. De la misma manera, si Klonoa se encuentra en un lugar estrecho, de tal manera que el bloque no cabe entre su cabeza y el techo, no podrá tomar el bloque. En la figura 3.21 se observan los bloques.

Figura 3.21: Bloques en Klonoa: Empire of Dreams.

Si Klonoa tiene un bloque encima de él, no podrá flotar. Klonoa puede realizar un doble salto si tiene un bloque, haciendo que el bloque sea lanzado hacia el suelo por debajo de Klonoa. Klonoa puede lanzar hacia la izquierda o derecha el bloque en cualquier momento.

Cuando Klonoa lanza el bloque en alguna dirección, este avanzará en línea

recta hasta chocar con una pared. Al chocar con una pared, el bloque caerá hasta el suelo. Si se sale y entra a un cuarto, los bloques regresarán a su posición original.

Existen en los escenarios tres tipos de interruptores, los interruptores temporales, los interruptores permanentes y los interruptores a presión. Cada interruptor está ligado a una sola cerradura. Por otro lado, existen cerraduras para los que se necesitan varios interruptores para hacerlo desaparecer. En general, todos los interruptores se encuentran en el mismo cuarto que su cerradura.

Los interruptores temporales y permanentes se encuentran flotando en el escenario. Estos dos interruptores son estéticamente idénticos. Ambos interruptores se activan con el ataque de Klonoa o al ser golpeados por un bloque.

Ambos tipos de interruptores permiten desaparecer ciertas cerraduras en alguna parte del escenario, pero en el caso de los interruptores temporales, las cerraduras volverán a aparecer después de un periodo de tiempo.

Los interruptores temporales se desactivan si se sale y entra de un cuarto, mientras que los interruptores permanentes no se desactivan. En la figura 3.22 se observan estos interruptores.

Figura 3.22: Interruptores y cerraduras en Klonoa: Empire of Dreams.

Los interruptores a presión se encuentran en el suelo. Estos interruptores se activan al tener encima un bloque, un enemigo o a Klonoa. Cuando no hay nada encima de estos interruptores, sus cerraduras aparecen instantáneamente. En la figura 3.23 se observa el funcionamiento de los interruptores a presión.

Figura 3.23: Interruptores a presión en Klonoa: Empire of Dreams.

En el suelo de los escenarios pueden encontrarse remolinos de viento. Si Klonoa camina por encima de un remolino de viento, se elevará hasta llegar al techo. Mientras Klonoa se encuentre subiendo por el remolino, no podrá moverse a la izquierda o a la derecha para escapar del remolino de forma común.

Si existe una plataforma o un bloque en medio de Klonoa y un remolino, el remolino no lo elevará. De igual manera, si Klonoa tiene un Bloque sobre el, el remolino no lo elevará. En la figura 3.24 se observa el funcionamiento de un remolino.

Figura 3.24: Remolino de viento en Klonoa: Empire of Dreams.

En el escenario pueden encontrarse pequeños globos con alas. Estos globos pueden ser atacados por Klonoa. Si Klonoa ataca a un globo, se sostendrá de él y puede quedar suspendido en el aire.

Si Klonoa está adherido a un globo, puede saltar para alcanzar más altura. Si Klonoa se encuentra en un remolino y un globo está a su alcance, puede

escapar del remolino atacando al globo y adhiriéndose a él. En la figura 3.25 se observa el funcionamiento de los globos con alas.

Figura 3.25: Globo con alas en Klonoa: Empire of Dreams.

En el videojuego original existen distintas clases de enemigos, en esta versión generalizada se utilizará solo uno. El enemigo que se utilizará se llama spiker.

Un spiker tiene una trayectoria o posición definida y son invulnerables al ataque de Klonoa. Cuando un spiker toca a Klonoa, Klonoa recibe daño de un golpe. Al recibir daño, Klonoa cuenta con un periodo corto de invencibilidad, en el cual puede atravesar enemigos. Este periodo de invencibilidad dura el suficiente tiempo para atravesar a un enemigo por completo sin recibir otro golpe.

Si un bloque se posiciona sobre un spiker, el spiker descenderá hasta el suelo. Los spiker se apilan, por lo que si un spiker cae sobre otro, se quedará encima de él. Si un spiker tiene un bloque encima y Klonoa lo quita, el spiker regresará a su posición o trayectoria original. En la figura 3.26 se observa el funcionamiento de los spiker con los bloques.

Figura 3.26: Enemigos spiker en Klonoa: Empire of Dreams.

Existen cerraduras de un solo sentido, los cuales Klonoa sólo puede atravesar en una dirección. De igual manera, los bloques sólo pueden moverse en la dirección hacia donde la cerradura les permita. Si un bloque es lanzado hacia el lado cerrado de la cerradura de un solo sentido, chocará como si fuera una pared. En la figura 3.27 puede observarse una puerta de un solo sentido.

Figura 3.27: Cerraduras de un solo sentido en Klonoa: Empire of Dreams.

En los escenarios pueden encontrarse puertas transportadoras. Una puerta transportadora tiene un número que la liga con otra puerta transportadora con el mismo número, la cual puede estar en otro cuarto. Sólo puede haber dos puertas con el mismo número.

Al entrar a una puerta transportadora Klonoa aparecerá en la otra puerta transportadora ligada. Entrar por una de estas puertas contará como salir de un cuarto y entrar a otro. En La figura 3.28 se observa una puerta transportadora.

Figura 3.28: Puerta transportadora en Klonoa: Empire of Dreams.

Para mayor información del videojuego y sus reglas, consultar[16].

3.2. Pruebas de dureza originales

Esta sección se dividirá en dos partes, al igual que con los videojuegos estudiados anteriormente, primero se explicarán las demostraciones sobre *NP-Dureza* y posteriormente la de *PSPACE-Completitud*.

En todas las demostraciones de esta sección, el problema de decisión a resolver es decidir si dado un nivel del videojuego generalizado presentado, es posible completarlo siguiendo sus reglas.

Al igual que en la sección de resultados de dureza conocidos, aunque en la mayoría de los casos no se menciona explícitamente, los escenarios creados tienen un tamaño polinomial, pues las estructuras que los componen tienen un tamaño constante o polinomial, y estas estructuras se repiten un número constante o polinomial de veces.

3.2.1. NP-Dureza

3.2.1.1. Crash Bandicoot

Para la demostración se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*, pero con algunas modificaciones. Por la construcción de los

componentes, no se necesita que se acceda a una sección de los componentes de cláusula desde los de variables, sólo se necesita que los componentes de variable activen a distancia el acceso para atravesar los componentes de cláusula correspondientes.

El componente de cruce no es necesario, pues no se accede físicamente desde la sección de las variables a la de las cláusulas. El componente de final sólo contiene la meta, no se requiere ningún estado especial al finalizar el nivel. Fuera de estos detalles, la construcción sigue las estructuras utilizadas en la técnica.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de inicio, el componente de variable y el componente de cláusula. Para mostrar la construcción de las secciones del nivel, se utilizará una vista aérea del mismo, aunque en el videojuego la vista se maneja con la cámara detrás del personaje.

La imágenes de estos componentes se crearon a partir de las texturas del videojuego, obtenidas de[26]. Todos los componentes se recorren de izquierda a derecha.

componente de inicio:

Para esta reducción se requiere que Crash muera al recibir un sólo golpe, por lo que hay que asegurarse que no tenga ningún Aku-Aku. En la figura 3.29 se muestra el componente de inicio, el cual obliga al jugador a perder todas las Aku-Aku.

En este componente, se encuentran tres Aku-Aku junto a una pared de *TNT*. El jugador tiene la opción de tomar las Aku-Aku para obtener invencibilidad y poder atravesar la pared de *TNT*, si el jugador decide no tomar alguna de las Aku-Aku, será destruida por la explosión de las *TNT*.

A una distancia mayor a la que se puede recorrer antes de que la invencibilidad se termine, hay otras dos paredes de *TNT* separadas de tal manera que sus explosiones no se afectan entre sí, de tal manera que la única forma de atravesarlas es atacar a las paredes y perder ambas Aku-Aku.

Es importante resaltar que si el jugador decide no tomar suficientes Aku-Aku para ser invencible al principio del componente, no hay forma de que pueda destruir las otras dos paredes de *TNT* sin morir.

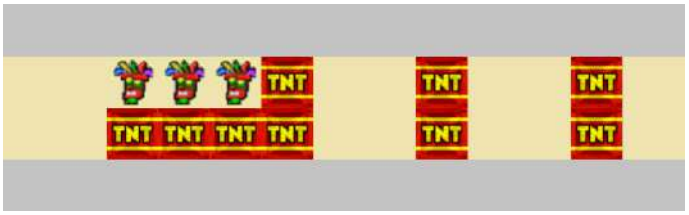


Figura 3.29: Componente de inicio de Crash Bandicoot para la estrctura de *3-SAT*.

Componente de variable:

En la figura 3.30 se muestra el componente de variable. La sección gris es una pared que separa el cuarto en dos secciones. Las partes cafés corresponde a suelo. las *TNT* se encuentran a nivel del suelo, de tal manera que se puede caminar sobre ellas llegando desde una zona de suelo sin necesidad de saltar.

En la línea de *TNT* de hasta la derecha hay una pared hecha de cajas *TNT*, de tal manera que no se puede avanzar hasta destruir esa pared de *TNT*. Debajo de las *TNT* hay vacío. Las cajas activadoras se encuentran en el aire, a una altura tal que Crash puede caminar entre la caja activadora y las *TNT*.

En cada camino puede activarse la caja activadora saltando por debajo de ella o saltando al lado y atacando, en cualquiera de los dos casos, al caer del salto se activarán las cajas de *TNT*. Si las cajas de *TNT* se activan, no habrá tiempo suficiente para llegar hasta la otra caja activadora, además las cajas *TNT* que se encuentran debajo de ambas cajas activadoras se destruirán por la explosión en cadena, por lo que si se elige una, la otra quedará inaccesible.

Al activar una caja *TNT* de debajo de la caja activadora, hay tiempo suficiente para llegar a la zona de suelo de la parte que se encuentra arriba de la caja activadora, en esa zona hay suficiente espacio para que Crash espere y la explosión de las *TNT* no lo eliminen, para poder seguir avanzando.

Hay un componente de variable por cada una de las variables de la fórmula, la caja activadora de arriba corresponde a la versión positiva de la variable y la caja activadora de abajo corresponde a la versión negada. La caja activadora activa cajas en cada componente de cláusula que contiene

a la literal, la manera en que estas cajas se activan se explicarán en el componente de cláusula.

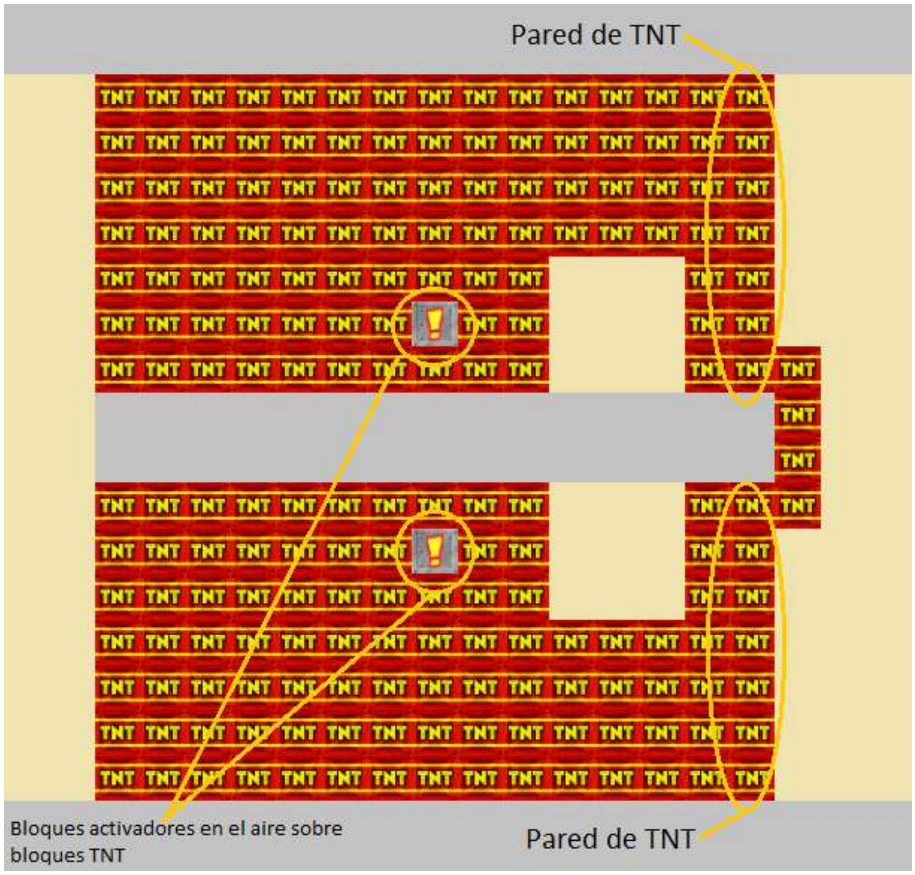


Figura 3.30: Componente de cariable de Crash Bandicoot para la estrctura de *3-SAT*.

Componente de cláusula:

En la figura 3.31 se muestra el componente de cláusula. La sección negra corresponden a vacío, mientras que la sección café representa suelo. Las cajas inactivas están a nivel del suelo, y al ser activadas se transforman en cajas de metal.

Las cajas con número 1 se activan con la caja activadora que se encuentra en la primera literal, la caja con número 2 se activan con la caja activadora de la segunda literal y las de número 3 se activan con la caja activadora de la tercera literal. Las dos secciones de piso están separadas a una distancia mayor a la que Crash puede saltar.

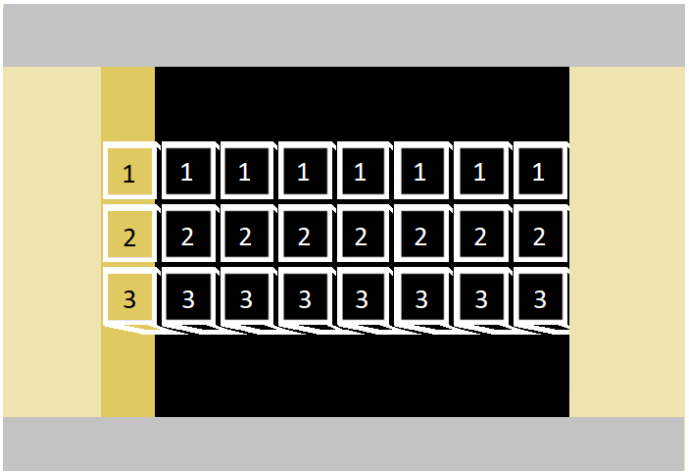


Figura 3.31: Componente de cláusula de Crash Bandicoot para la estrctura de *3-SAT*.

De esta manera se puede superar cada componente de cláusula si y sólo si se activó al menos una de sus literales.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un nivel de Crash Bandicoot generalizado, es posible completarlo siguiendo sus reglas es *NP-Duro*.

Todos los elementos presentados en esta demostración se encuentran también en la secuela, Crash Bandicoot 2: Cortex Strike Back. Por esta razón, esta demostración es válida también para este videojuego. En esta secuela se agrega una habilidad que permite a Crash saltar una mayor distancia, pero los componentes pueden adecuarse para considerar esta distancia sin problema.

Para las secuelas Crash bandicoot 3: Warped y Crash Bandicoot: The Wrath of Cortex, se agregaron varias habilidades más. Las más importantes que afectarían a la demostración son: una habilidad que permite correr a mayor velocidad, un doble giro que permite caer más lentamente para saltar más distancia, un doble salto y una bazooka que permite disparar a distancia a objetos en línea recta desde la posición de Crash sin obstáculos intermedios.

Para las primeras tres habilidades, y al igual que con la secuela explicada anteriormente, los componentes pueden adaptar sus distancias para considerar los nuevos valores de velocidad y distancia de salto de Crash. Para

la última habilidad, sería necesario modificar el componente de variable presentado ligeramente.

Esta modificación consiste en posicionar cajas metálicas alrededor de los activadores, de tal manera que Crash no pueda disparar a los mismos desde una posición que no sea por debajo de la misma. Este componente modificado se muestra en la figura 3.32.

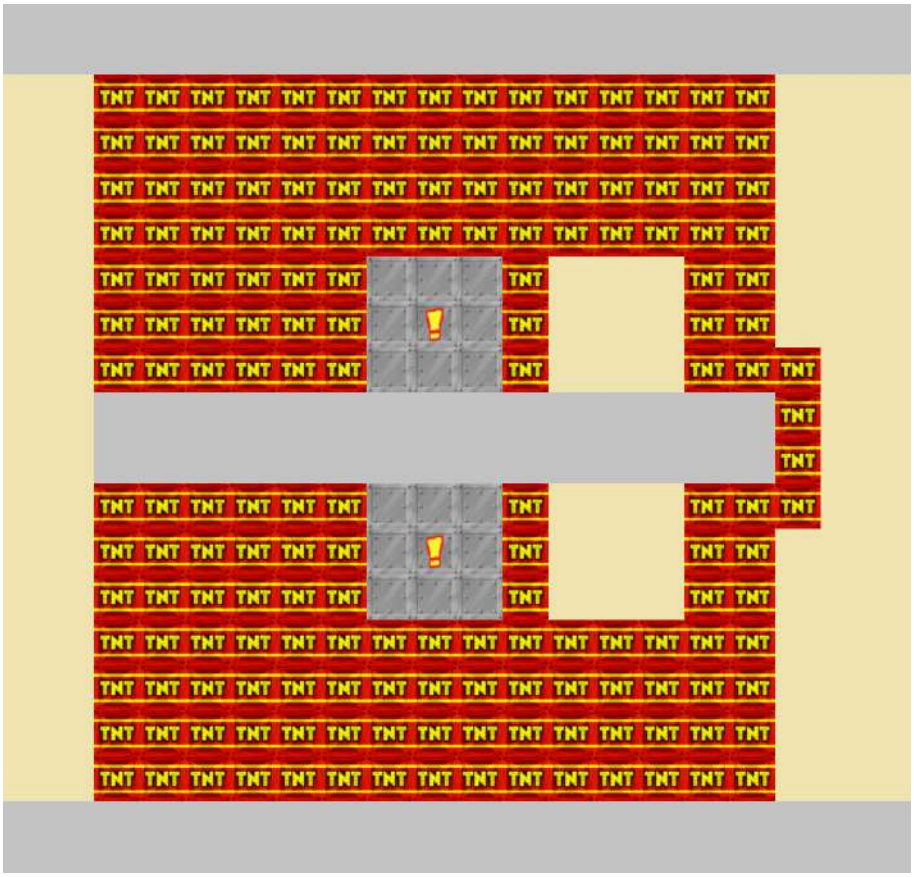


Figura 3.32: Componente de variable de Crash Bandicoot para la estrctura de *3-SAT* modificado para las secuelas.

Dada esta construcción, puede concluirse que es posible definir versiones generalizadas de estas secuelas con un problema de decisión *NP-Duro* análogo al presentado en esta sección.

3.2.1.2. The Binding of Issac: Rebirth

Para demostrar la *NP-Dureza* de The Binding of Isaac: Rebirth generalizado, basta con aplicar el metateorema 3 mencionado en [27]. Este metateorema dice lo siguiente:

Un juego es *NP-Duro* si contiene puertas y caminos de un solo sentido, y sucede alguna de las siguientes afirmaciones:

- El juego contiene llaves recolectables y secciones obligadas a ser recorridas.
- El juego contiene llaves acumulables y secciones obligadas a ser recorridas.
- El juego contiene llaves recolectables y acumulables y el personaje debe llegar a una salida.

Dado que The Binding of Issac: Rebirth cuenta con puertas, caminos de un solo sentido, llaves recolectables y acumulables, y el personaje debe llegar a una salida, se sigue que es *NP-Duro*.

Por otro lado en este trabajo se presenta una demostración alternativa, utilizando algunos elementos extra explicados en la sección de reglas.

Para esta demostración se utilizará la idea de la técnica para *NP-Dureza* de reducción de *3-SAT*, pero con algunas modificaciones. En la sección de activación de cada componente de cláusula se encuentran tres llaves, sin embargo, para abrir el camino hacia una, se requiere cerrar permanentemente el acceso a las otras dos, por lo que sólo se puede obtener una llave por cláusula.

En la sección de revisión, se encuentra una puerta cerrada por cada cláusula, por lo que si bien cada revisión de cláusula no necesita que su propia cláusula sea activada, si se necesita que se hayan activado todas las cláusulas para atravesar la sección de revisión completa.

El componente de final sólo contiene la meta, no se requiere ningún estado especial al finalizar el nivel. Fuera de estos detalles, la construcción sigue la estructura utilizada en la técnica.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de inicio, componente de cruce, componente de variable y componente de cláusula. Para mostrar la construcción de las secciones del nivel, se utilizará una vista aérea del mismo, al igual que en el videojuego original. Las imágenes de los componentes se crearon a partir de imágenes del videojuego obtenidas de[25]. Todos los componentes se recorren de izquierda a derecha.

Componente de inicio:

Para esta demostración, se requiere que Isaac muera al recibir un sólo golpe. Para lograr esto, se construirá un cuarto que contenga 11 corazones de alma y 12 corazones. Los 12 corazones le permiten a Isaac llenar todos los contenedores de corazón que tenga, sin importar la cantidad, pues el máximo son 12. Los 11 corazones de alma le permiten llegar al máximo de contenedores de corazón si es que llega con una cantidad menor al máximo. De esta forma, dado que cada corazon permite dos golpes, y sin importar la cantidad de corazones que Isaac tenga al momento de entrar a este cuarto, se le permite tener un total de 24 golpes antes de morir. Este cuarto se observa en la parte izquierda de la figura 3.33.

Posteriormente, se construirán un cuarto con un pasillo muy largo. Este pasillo contiene 23 secciones de púas, separadas por una distancia mayor a la que Isaac puede recorrer en el tiempo que dura su invencibilidad después de recibir daño. De esta manera, la única forma de superar este cuarto es recibir 23 golpes, lo que dejará a Isaac a un golpe de morir. Se observa este cuarto en la parte superior de la figura 3.33, aunque en la figura solo se observan dos secciones de púas, las otra 21 púas se encuentran en los puntos suspensivos.

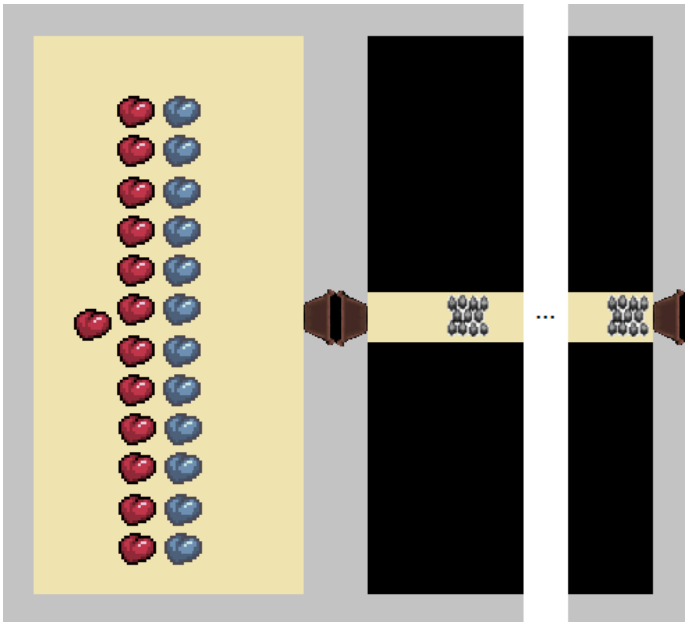


Figura 3.33: Componente de inicio de The Binding of Isaac: Rebirth para la estrctura de *3-SAT*, se muestran dos cuartos. El cuarto de la derecha contiene 23 púas.

Componente de cruce:

Para este componente se utilizará un cuarto de 15×9 cuadros. Dentro de este cuarto, hay cuatro secciones hechas con bloques de metal, acomodadas como se muestra en la figura 3.34.

La parte sombreada son bloques de metal de 3×6 secciones. La separación entre cada sección de bloques de metal es de un cuadro, así como la separación entre las secciones y la pared. Los elementos marcados con P_1, S_1, P_2 y S_2 son puertas, las cuales están abiertas desde el principio. Para obligar a que se siga una ruta específica, de tal manera que si se entra por P_1 sólo se pueda salir por S_1 y al entrar por P_2 sólo se pueda salir por S_2, se utilizarán enemigos wall hugger.

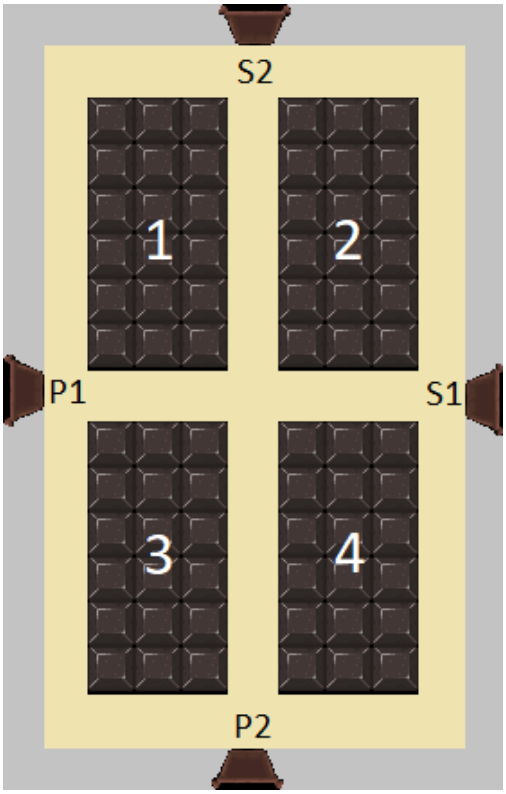


Figura 3.34: Cuarto del componente de cruce de The Binding of Isaac: Rebirth para la estrctura de *3-SAT*.

Los enemigos se colocarán en el cuarto en un principio como se observa en la figura 3.35 y se moverán alrededor de las secciones de bloques de metal.

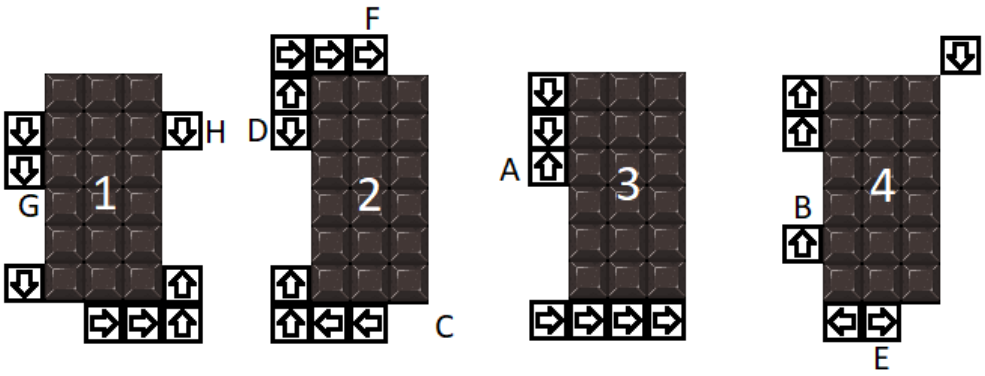


Figura 3.35: Enemigos del componente de cruce de The Binding of Isaac: Rebirth para la estrctura de *3-SAT*.

Los cuadros transparentes representan a los enemigos y las flechas su dirección. Todos los enemigos se mueven a la misma velocidad, igual o menor a la mitad de la velocidad de Isaac, de tal manera que el jugador puede seguir su dirección sin que un enemigo lo alcance por detrás. Se le llamará al tiempo que tarda un enemigo en cambiar de cuadro, un movimiento.

Para verificar que el componente funciona, se verá lo que sucede en el caso de entrar al cuarto a través de cada puerta. Si se entrara por S_1 o S_2, los enemigos que giran alrededor de las estructuras 2 o 4 tocarían y matarían a Isaac al entrar.

Si se entra por la puerta P_1, Isaac no puede ir hacia abajo, ya que el enemigo marcado con A no le permite avanzar en esa dirección, y si intenta seguir por esa dirección, eventualmente lo alcanzaría y lo eliminaría.

Isaac tampoco puede ir por arriba porque hay enemigos que se mueven hacia él que vienen de esa dirección. La única opción es moverse a la derecha, siguiendo la dirección de los enemigos que se mueven alrededor de la estructura 1 que rodean a Isaac. En cuatro movimientos, el espacio que inició en el cuadro donde inicia Isaac se encontrará en el centro.

Los enemigos que se mueven alrededor de la estructura 4 dejarán en el centro el espacio marcado con B y los de la estructura 2 dejan el espacio marcado con C en el centro y los bloques marcados con D y H justo arriba de Isaac, todos con el espacio suficiente para que Isaac pueda llegar hasta ahí sin ser aplastado.

El bloque marcado con D impide que el jugador se mueva hacia arriba, los enemigos de la estructura 4 impiden ir hacia abajo, por lo que sólo se puede seguir el camino hasta S_1. Llegando a S_1, el enemigo marcado con E se encuentra justo abajo y F justo arriba, por lo que sólo se puede entrar a S_1.

Si se entra por P_2, no se puede ir hacia la izquierda o a la derecha, pues hay enemigos que vienen de esas direcciones. Esto obliga a seguir la dirección de los enemigos que se mueven alrededor de la estructura 4.

En siete movimientos, el espacio que inició en el cuadro donde inicia Isaac se encontrará en el centro, los enemigos que se mueven alrededor de la estructura 1 dejarán el espacio marcado con G al centro, de tal manera que no se puede ir a la izquierda por estos enemigos.

Si va a la derecha siguiendo los enemigos de la estructura 4, eventualmente se encuentra con E (en tres movimientos). Por lo anterior Isaac debe ir hacia arriba, siguiendo a los enemigos que se mueven alrededor de la estructura 1.

En ese momento el enemigo marcado con H se encuentra a la izquierda del centro y el enemigo marcado con D se encuentra a la derecha del centro, por lo que no causará problemas.

Al llegar a S_2, si se trata de seguir el camino a la izquierda, se encontrará con el enemigo marcado con H, y se encontrará con D si trata de ir a la derecha, por lo que sólo se puede entrar a la puerta S_2.

Componente de variable:

El componente de variable se compone de tres cuartos. Los dos cuartos de la izquierda sirven como entradas que conectan con las salidas del componente de variable anterior.

Cada puerta de salida de este cuarto inicia con un wall hugger enfrente. Se puede entrar a esta puerta esperando que el wall hugger se mueva. Si se intenta regresar por la puerta, el wall hugger golpeará a Isaac.

El segundo cuarto corresponde a la decisión de la variable. En este cuarto se encuentran dos puertas de un solo sentido de salida que funcionan de la misma forma que las puertas de los cuartos anteriores.

Este componente se observa en la figura 3.36.

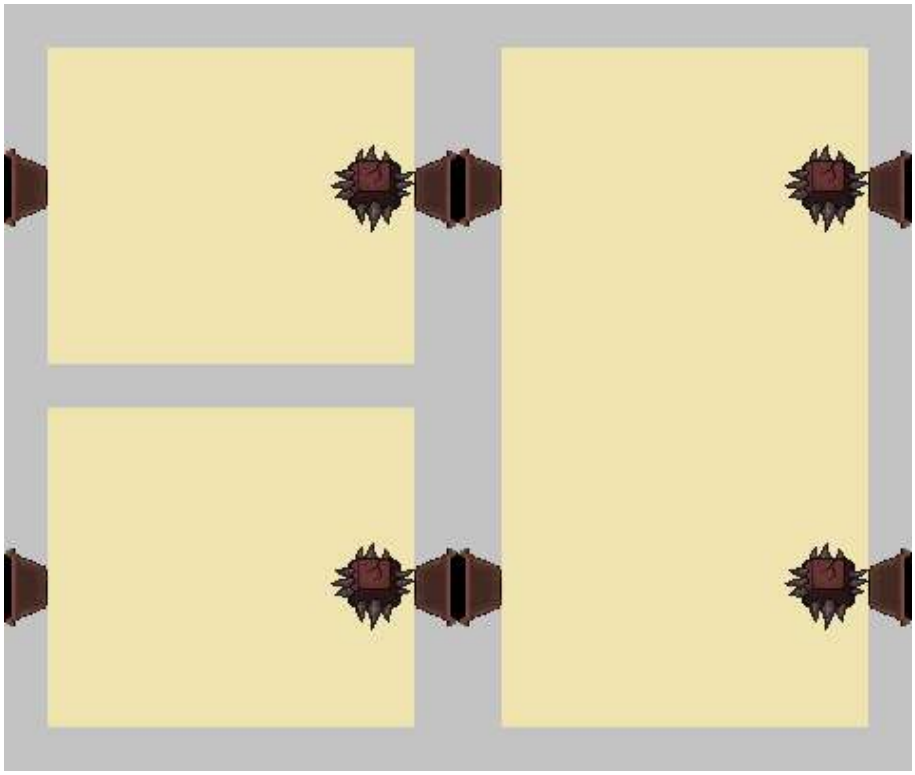


Figura 3.36: Componente de variable de The Binding of Isaac: Rebirth para la estrctura de *3-SAT*.

Componente de cláusula:

Para la sección de activación de este componente se tienen tres entradas, una por cada variable de la cláusula. La idea general es que cada entrada está separada de las demás, de tal manera que te permita tomar una llave al final del camino de esa entrada, pero que para tomarla se cierre el camino para ya no poder tomar la llave en las otras entradas. Para esto se utilizará una estructura como la que se observa en la figura 3.37.

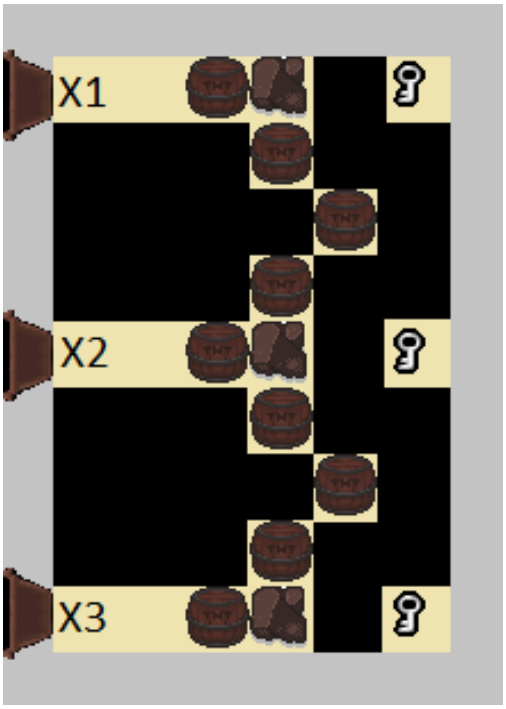


Figura 3.37: Sección de activación del componente de cláusula de The Binding of Isaac: Rebirth para la estrctura de *3-SAT*.

Para llegar a la llave, por ejemplo, entrando desde la puerta x_1, es necesario crear un puente. Para crear un puente es necesario destruir la TNT, la cual a su vez creará una explosión. La explosión de la TNT destruirá la piedra que tiene enfrente y creará un puente. Además, la explosión destruirá la TNT que está en diagonal a su derecha, y creará una explosión en cadena con todas las TNT.

Como la explosión que destruye a las piedras no viene de la izquierda, no creará un puente, pero si las destruirá, por lo que aunque se entre por las otras entradas, no se podrá tomar otra llave de este cuarto. Esto mismo sucede en los otros dos caminos, el de la puerta x_2 y el de la puerta x_3, por lo que al llegar a este cuarto se puede tomar exactamente una llave.

Como se explicó al principio de la demostración, la sección de revisión de cada cláusula es simplemente un cuarto con una puerta cerrada con llave necesaria para poder avanzar. Esta sección del componente se observa en la figura 3.38.

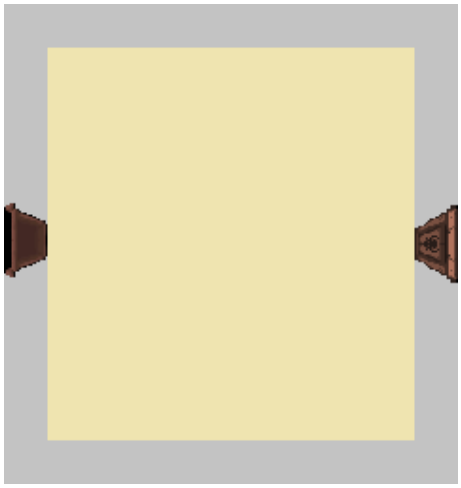


Figura 3.38: Sección de revisión del componente de cláusula de The Binding of Isaac: Rebirth para la estrctura de *3-SAT*.

De esta manera se pueden superar todos los componentes de cláusula si y sólo si se activó al menos una de las literales de cada uno de ellos.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un nivel de The Binding of Isaac: Rebirth generalizado, es posible completarlo siguiendo sus reglas es *NP-Duro*.

3.2.1.3. Sheep Raider

Para esta demostración se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el componente de inicio no es necesario, pues no se requiere ningún estado específico durante el trayecto de la construcción.

El componente de final sólo contiene la meta. Las reglas del videojuego obligan a que para terminar el nivel se lleve una oveja a la meta.

Finalmente, el componente de cruce no es necesario pues se trata de un videojuego en 3D que te permite crear caminos cerrados a distintas elevaciones.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de variable y componente de cláusula. Para mostrar la construcción de las secciones del nivel, se utilizará una vista aérea

del mismo, Aunque en el videojuego se presenta una cámara en tercera persona controlada por el jugador.

Todos los componentes se recorren de izquierda a derecha. Al ser un videojuego en 3D, y no existir un editor de niveles, no fue posible utilizar imágenes de elementos del videojuego para crear los componentes. Por lo tanto, se utilizaron imágenes de elementos de otro videojuego, obtenidas de[25].

Componente de variable:

El componente de variable es sencillo, basado en el utilizado en la demostración de *NP-Dureza* de Portal que utiliza la técnica de reducción a *3-SAT*. Este componente consiste en una sección del nivel que se divide a dos caminos disjuntos.

Ambos caminos se conectan a una superficie con menor elevación que el camino de donde llega el jugador, con una caída larga. Ambos caminos están separados con la suficiente distancia para que el lobo no pueda llegar de un salto de una a otra. Este componente puede verse en la figura 3.39.

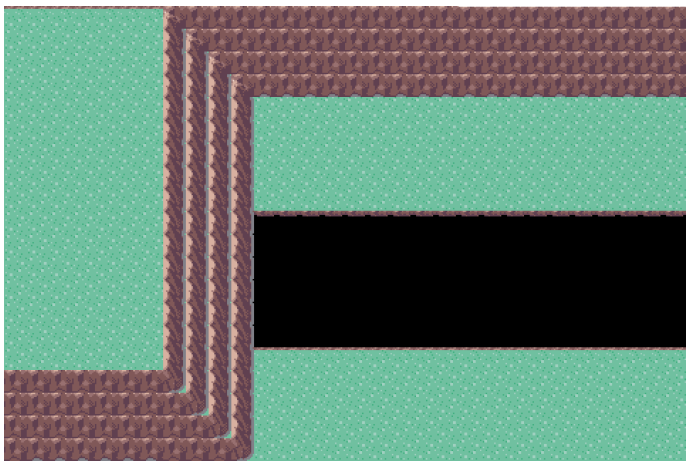


Figura 3.39: Componente de variable de Sheep Raider para la estrctura de *3-SAT*.

Al final del último componente de variable, se encuentra una oveja y un precipicio que te deja caer en la sección de revisión del primer componente de cláusula. Este precipicio tiene la suficiente altura para no poder regresar al caer al componente de cláusula.

Componente de cláusula:

La sección de activación del componente de cláusula se encuentra a mayor elevación que la sección de revisión. La diferencia de elevación no permite ir de la sección de revisión a la de activación.

Si bien se puede acceder de la sección de activación a la de revisión, dado que la oveja está a final del último componente de cláusula, no se podrá terminar el nivel si se cae de la sección de activación a la de revisión. En la figura 3.40 se observa el componente de cláusula.

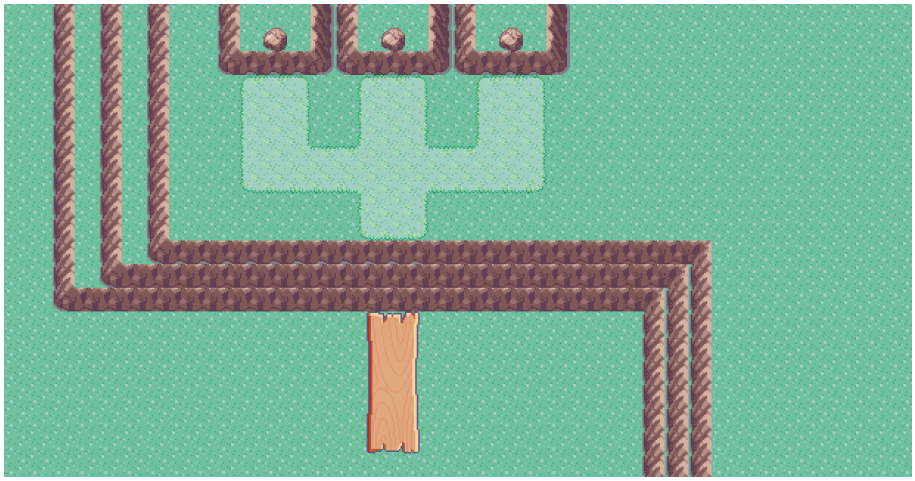


Figura 3.40: Componente de cláusula de Sheep Raider para la estrctura de *3-SAT*.

La sección de activación del componente de cláusula se compone de tres entradas. En cada una de estas entradas se encuentra una piedra de que el lobo puede empujar. La piedra al ser empujada cae en una plataforma de la sección de revisión. Cada plataforma está separada con suficiente distancia para que el lobo no pueda llegar de un salto de una a otra.

En la sección de revisión, se tienen dos partes a distintas elevaciones. Se llega por la parte izquierda del componente, a la sección menos elevada.

La parte más elevada puede ser alcanzada desde la parte menos elevada desde un costado con plataformas en forma de escalera con dobles saltos, pero no con saltos normales. De esta manera el lobo puede subir sólo, pero no puede subir si tiene una oveja. En la parte a menor elevación se encuentra una catapulta, en la cual se puede dejar una oveja.

Las piedras de la parte de activación caen en la parte de mayor elevación de la sección de revisión. Debajo de las plataformas de la sección de activación hay un espacio vacío, de tal manera que el lobo puede entrar para empujar la piedra cuando cae. Un ejemplo de este tipo de plataformas se muestra en la figura 3.41.

Figura 3.41: Captura de Sheep Raider con plataforma elevada. Debajo de ella hay un espacio vacío en el que podría entrar el lobo para empujar una piedra hacia el lado contrario de la plataforma si la piedra cayera de ella.

La trayectoria de las piedras permite dejarlas caer de un lado de la catapulta. De esta manera, la oveja puede ser lanzada a la parte de mayor elevación si y sólo si se tiene al menos una piedra en la parte elevada de la sección de revisión.

Finalmente, la parte de mayor elevación de la sección de revisión se conecta con la sección de revisión del siguiente componente de cláusula, o con el componente de final en el caso del último componente de cláusula.

De esta manera se puede superar cada componente de cláusula si y sólo si se activó al menos una de sus literales.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un nivel de Sheep Raider generalizado, es posible completarlo siguiendo sus reglas es *NP-Duro*.

3.2.1.4. Klonoa: Empire of Dreams

Como Klonoa: Empire of Dreams generalizado cuenta con interruptores temporales, posiblemente podría utilizarse la técnica de reducción de ciclo hamiltoniano en una gráfica en cuadrícula, pero este videojuego no es en 3D ni en vista aérea.

Para la demostración mencionada se requeriría una forma de crear pasillos que vayan hacia abajo y hacia arriba con un mismo costo de tiempo. Esta demostración no se presenta en este trabajo.

Sin embargo, es posible demostrar la *NP-Dureza* de este videojuego sin utilizar puertas con botones temporales. Esta demostración se presenta a continuación.

Para esta demostración se utilizará la técnica para *NP-Dureza* de reducción de *3-SAT*. En este caso el componente de cruce no es necesario, ya que pueden evitarse los cruces con el uso de puertas de teletransporte.

El componente de inicio no es necesario, pues no se requiere ningún estado específico durante el trayecto de la construcción. El componente de final contiene simplemente la meta junto con las tres estrellas necesarias para terminar el nivel.

Algunos de los componentes presentados dependen de la idea de que al pasar por él una segunda vez, los bloques tendrán su posición reiniciada, por lo que cada componente es un cuarto separado.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de inicio, componente de variable y componente de cláusula. Las imágenes de los componentes se crearon a partir de capturas del videojuego. Todos los componentes se recorren de izquierda a derecha.

Componente de variable:

Este componente consiste en dos entradas y dos salidas. Sin importar por cual entrada se llegue, no será posible regresar.

Enmedio de las dos salidas se encuentra un globo con alas permite llegar a la salida superior.

En cada camino que conduce a una salida, se encuentra una puerta de un solo sentido. Esta puerta impide regresar una vez elegida una salida. El componente de variable puede observarse en la figura 3.42.

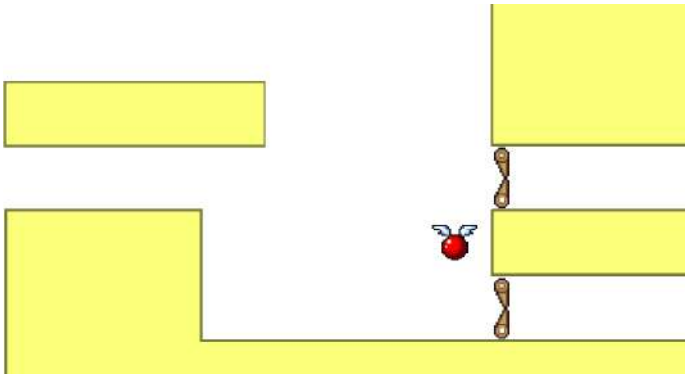


Figura 3.42: Componente de variable de Klonoa: Empire of Dreams para la estrctura de *3-SAT*.

Componente de cláusula:

El componente de cláusula puede observarse en la figura 3.43. En la sección de activación de este componente tiene tres entradas a la izquierda a distintas elevaciones. Estas entradas conectan con los componentes de variables correspondientes a las literales de la cláusula.

En cada entrada se encuentra un bloque, el cual puede ser tomado desde el lado de la entrada. del otro lado del bloque hay techo que no permite tomar el bloque desde este lado.

A la derecha de las entradas y sus bloques se encuentran un remolino junto a tres globos con alas. Los globos te permiten elegir a qué altura escapar del remolino.

Independientemente de que se pueda elegir el camino de cualquier entrada, solo se podrá regresar por el camino por el que se llegó, por la configuración de los bloques.

Finalmente, pasando el remolino se encuentra un interruptor, que abre permanentemente una cerradura.

La sección de activación consiste en un pasillo con una cerradura enmedio. Esta cerradura se abre con el interruptor de la sección de activación.

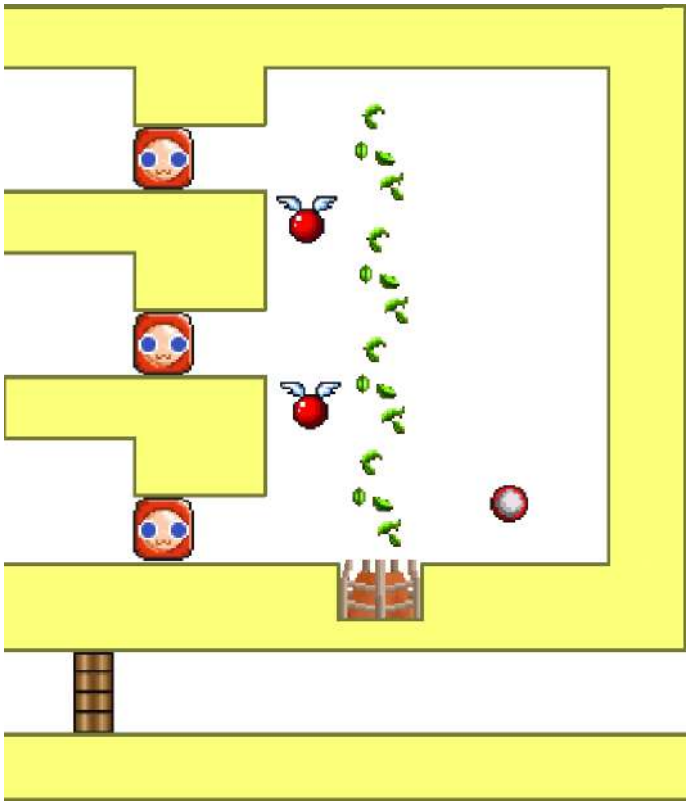


Figura 3.43: Componente de cláusula de Klonoa: Empire of Dreams para la estrctura de *3-SAT*.

De esta manera se puede superar cada componente de cláusula si y sólo si se activó al menos una de sus literales.

A partir de esta construcción, y utilizando la técnica mencionada, se puede concluir que decidir si dado un nivel de Klonoa: Empire of Dreams generalizado, es posible completarlo siguiendo sus reglas es *NP-Duro*.

Todos los elementos presentados en esta demostración se encuentran también en las secuelas, Klonoa: Dream Champ Tournament y Kaze no Klonoa: Moonlight Museum. Por esta razón, puede concluirse que es posible definir versiones generalizadas de estas secuelas con un problema de decisión *NP-Duro* análogo al presentado en esta sección.

3.2.2. PSPACE-Dureza y PSPACE-Completitud

3.2.2.1. Klonoa: Empire of Dreams

Para probar la *PSPACE-Completitud* del problema, es necesario probar su *PSPACE-Dureza* y pertenencia en *PSPACE*.

El problema de decidir si es posible terminar un nivel de Klonoa: Empire of Dreams generalizado siguiendo sus reglas se encuentra en *PSPACE*, pues el videojuego cumple las características requeridas en la técnica de pertenencia en *PSPACE* explicada anteriormente.

Para demostrar la *PSPACE-Dureza* del problema, se utilizará la técnica para *PSPACE-Dureza* de reducción de *TQBF*.

Al contrario de la prueba de *NP-Dureza* de este videojuego, esta prueba depende de que los bloques conserven los cambios en su posición durante la ejecución. Por lo tanto, para esta demostración, todos los componentes y la estructura en general se encuentran en un mismo cuarto.

Una consecuencia del uso de un solo cuarto para la demostración, es la necesidad de un componente de cruce, pues las puertas de teletransporte tienen el efecto de cambio de cuarto, por lo que reinician la posición de los bloques.

Por otro lado, en esta demostración se utilizarán los interruptores con puertas temporales.

Al igual que en la prueba de *NP-Dureza*, el componente de final contiene simplemente la meta junto con las tres estrellas necesarias para terminar el nivel.

Una vez explicada la estructura de la reducción, basta con explicar la construcción del componente de inicio, componente de cruce y componente de puerta. Al igual que con la demostración de *NP-Dureza* de este videojuego, las imágenes de lo componentes se crearon a partir de capturas del videojuego.

Componente de inicio:

Para esta demostración se requiere que Klonoa muera al recibir un solo golpe. Para llegar a este estado, utilizaremos un componente hecho con un pasillo con el espacio suficiente para que Klonoa pueda atravesarlo caminando, pero sin posibilidad de saltar.

El componente cuenta primero con dos corazones, de tal manera que Klonoa pueda obtener el máximo de vida posible.

Posteriormente, hay dos spiker, separados con una distancia mayor a la que se puede recorrer antes de que el periodo de invencibilidad se termine. De esta manera, la única forma de atravesar el componente es perdiendo dos puntos de vida. Este componente puede observarse en la figura 3.44.

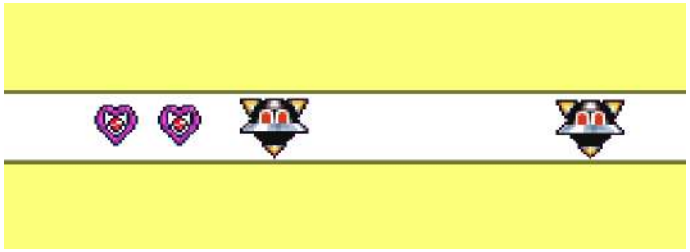


Figura 3.44: Componente de inicio de Klonoa: Empire of Dreams para la estructura de *TQBF*.

Componente de cruce:

El componente de cruce puede verse en la figura 3.45. Para este componente se tienen dos entradas a distinta elevación con un interruptor temporal en cada una. Cada interruptor abre una cerradura cruzada, el interruptor de la entrada de arriba abre la cerradura de abajo y el interruptor de la entrada de abajo abre la cerradura de arriba. Los interruptores tienen la duración suficiente para cruzar su cerradura.

Pasando los interruptores, se encuentra una puerta de un solo sentido, de tal manera que no pueda pasarse de una entrada a otra, o activar ambos interruptores.

Posteriormente se encuentran dos salidas, a distintas alturas, bloqueadas con las cerraduras mencionadas anteriormente. Se tiene un globo con alas en medio de las dos cerraduras que permite llegar a la cerradura superior.

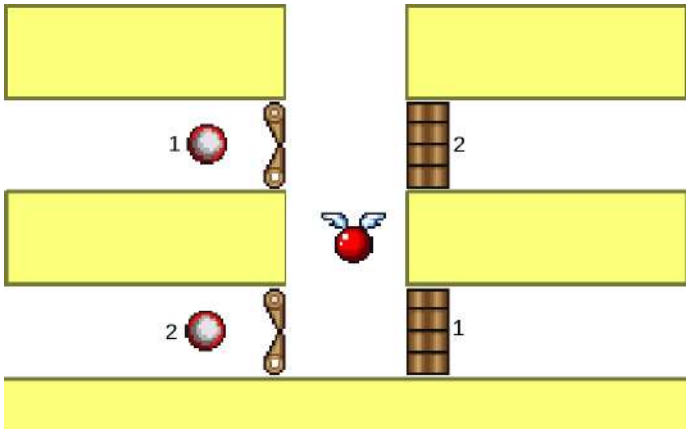


Figura 3.45: Componente de cruce de Klonoa: Empire of Dreams para la estructura de *TQBF*.

Componente de puerta:

El componente de puerta puede observarse en estado abierto en la figura 3.46. En los caminos para abrir y cerrar, se entra por la parte inferior y se sale por la parte superior. Estos dos caminos se encuentran separados por un remolino. Si se intenta cruzar de un camino al otro, Klonoa quedará atrapado en el remolino.

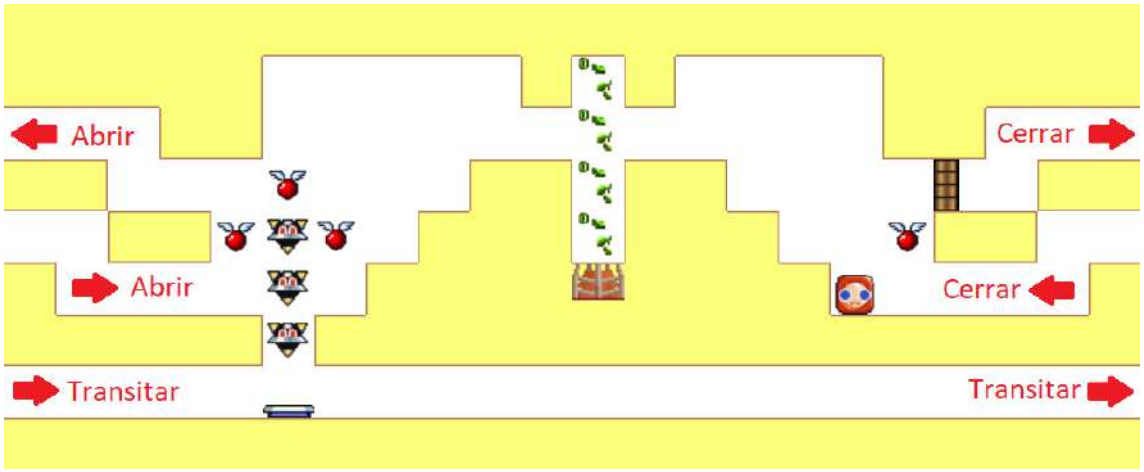


Figura 3.46: Componente de puerta de Klonoa: Empire of Dreams para la estructura de *TQBF*.

El remolino se encuentra rodeado por ambos lados con paredes que impiden a Klonoa llegar a él con el bloque. En ambos caminos se encuentran varios

globos con alas que permiten llegar a las salidas. Los globos con alas del camino para abrir permiten además moverse por la sección sin ser golpeado por los spiker.

El hecho de que la estrctura se construya en un solo cuarto, en principio permite llevar un bloque de un componente de puerta a otro. Para evitar eso, las entradas y salidas de las zonas donde hay un bloque tienen una forma de escalera.

Si Klonoa carga un bloque no puede salir por ninguno de los caminos, pues son caminos estrechos. Por otro lado, si Klonoa lanza el bloque hacia alguna de las salidas, este chocará con la pared y bloqueará la salida, dejando este camino bloqueado para el resto de la ejecución.

En el caso del camino para cerrar, la salida está bloqueada con una cerradura. Esta cerradura está asociada al interruptor de presión que se encuentra en el camino para transitar. La única forma de abrir esta cerradura desde el camino para cerrar, es haciendo que un spiker presione el interruptor haciéndolo bajar.

Para hacer que los spiker bajen, es necesario lanzar el bloque por la separación de los caminos para abrir y para cerrar. Este bloque chocará con la pared, caerá sobre los spiker y los hará caer. Al lanzar el bloque y hacer bajar a los spikers, estos bloquearán el camino para transitar.

Por otro lado, desde el camino para abrir, si el componente se encuentra en estado cerrado, basta con quitar el bloque para dejar el camino para transitar desbloqueado. Klonoa no puede bajar al camino para transitar desde el camino para abrir, ya que de intentarlo, los spikers lo golpearían y lo matarían.

Una vez que Klonoa ha levantado el bloque, puede lanzarlo al camino para cerrar, posteriormente puede utilizar los globos con alas alrededor de los spiker para llegar a la salida del camino para abrir.

Si bien puede decidirse no lanzarse el bloque hacia el camino para cerrar desde el camino para abrir, esto impediría avanzar en un futuro al pasar de nuevo por el camino para cerrar, por lo que se puede considerar que la puerta no está realmente abierta hasta pasar el bloque del otro lado.

Por la observación mostrada en la sección de la técnica utilizada, dejar

a la decisión del jugador abrir o no más posibilidades de avanzar, en las que tomar la decisión positiva no cierra ningún camino, no afecta en la construcción.

Finalmente, el camino para transitar es un pasillo simple en el que no se puede saltar. Si el componente está en estado abierto, no habrá nada que impida avanzar. Por otro lado si el componente está en estado cerrado, los spiker bloquean el camino y no se puede avanzar sin recibir daño.

A partir de esta construcción, y utilizando la técnica mencionado, se puede concluir que decidir si dado un nivel de Klonoa: Empire of Dreams generalizado, es posible completarlo siguiendo sus reglas es *PSPACE-Completo*.

Al igual que en la demostración de *NP-Dureza* de este mismo videojuego, todos los elementos presentados en esta demostración se encuentran también en las secuelas, Klonoa: Dream Champ Tournament y Kaze no Klonoa: Moonlight Museum. Por esta razón, puede concluirse que es posible definir versiones generalizadas de estas secuelas con un problema de decisión *PSPACE-Completo* análogo al presentado en esta sección.

3.3. Resumen

En este capítulo se presentan problemas de decisión en videojuegos que no han sido estudiados en trabajos anteriores. Como resultado se obtuvieron cuatro demostraciones de *NP-Dureza* y una de *PSPACE-Completitud*.

Al igual que en el capítulo anterior, en principio se explican las reglas generales de cada uno de los videojuegos estudiados, en una versión generalizada que pueda ser estudiada como un problema discreto.

Posteriormente se presentan las demostraciones de *NP-Dureza* y *PSPACE-Completitud* originales estudiadas en este trabajo. Dado que todas las demostraciones utilizan una técnica, se clasifican dependiendo de si se demuestra dureza o completitud en *NP* o en *PSPACE*.

4 Conclusiones y trabajo a futuro

4.1. Conclusiones

En este trabajo se recopilaron estudios de la complejidad de problemas de decisión de ocho videojuegos presentados en trabajos previos.

Estos videojuegos son Super Mario Bros., The Legend of Zelda: A Link To The Past, Donkey Kong Country, Pokémon Rojo/Azul, Metroid, Mario Kart 64, Portal y Tetris.

Por otro lado, los artículos en donde se presentan originalmente los estudios de complejidad de estos videojuegos y que fueron retomados en este trabajo son [1], [4], [8], [9] y [5].

Usando como base las demostraciones presentadas en esos trabajos, y las técnicas presentadas principalmente en [1], se lograron desarrollar las demostraciones originales que se presentan en el capítulo tres de este trabajo.

Como resultado, se analizaron cuatro videojuegos de los cuales no existe un trabajo previo que analice su complejidad. Estos videojuegos son Crash Bandicoot, The Binding of Isaac, Sheep Raider y Klonoa: Empire of Dreams.

Para los primeros tres videojuegos se presenta una demostración de *NP-Dureza*, mientras que para el último se presenta una de *NP-Dureza* y una de *PSPACE-Completitud*.

Finalmente, para Crash Bandicoot y Klonoa: Empire of Dreams, se menciona que sus respectivas secuelas comparten los suficientes elementos para que las demostraciones mostradas puedan aplicarse a ellos.

En el caso de Crash Bandicoot, se presentan las modificaciones necesarias

de la reducción mostrada para aplicarla a sus secuelas, mientras que para Klonoa: Empire of Dreams, la reducción puede aplicarse directamente.

4.2. Trabajo a futuro

En algunos de las fuentes utilizadas como referencia en este trabajo, se presentan problemas abiertos.

En las demostraciones de *NP-Dureza* y *PSPACE-Completitud* de Mario Kart 64, presentadas en[4], se utiliza una conjetura para calcular el tiempo óptimo neceario para recorrer una pista. En[4] se deja como trabajo a futuro la comprobación de que es posible realizar este cálculo utilizando la física del juego original.

En la demostración de *NP-Dureza* de Super Mario Bros. presentada en[9], se requiere del uso de una codificación que permita guardar m veces consecutivas un elemento o nivel utilizando $O(\log m)$ bits. Se deja como trabajo a futuro en[9] el análisis de la completitud del juego sin el uso de esta codificación.

En esta misma demostración se utilizan varios niveles para la construcción. En[9] se deja también como trabajo a futuro el análisis de completitud de Super Mario Bros. generalizado utilizando un sólo nivel.

En[5] se menciona que si bien se presenta una demostración para una versión generalizada de Tetris, la cual es presentada aquí, puede analizarse la complejidad del videojuego tomando en cuenta otras cuestiones del mismo. Entre estas cuestiones se encuentra el análisis de la complejidad de Tetris con un tablero vacío, o tomando un tablero de tamaño constante.

De los videojuegos presentados aquí, solamente uno tiene prueba de completitud, Klonoa: Empire of Dreams. Para los otros juegos, haría falta determinar si se encuentran en *NP*, o si se puede encontrar una reducción de un problema *PSPACE-Duro* a sus problemas de decisión.

En la demostración de la *NP-Dureza* de Klonoa: Empire of Dreams, se menciona que podría aplicarse la técnica de ciclo hamiltoniano en una gráfica en cuadrícula, con algunas modificaciones. Esta demostración no se presenta en este trabajo, por lo que queda como trabajo a futuro.

Finalmente, existen muchos videojuegos cuya complejidad no ha sido estudiada, en realidad, cada año aparecen una gran cantidad de videojuegos nuevos con reglas distintas. Por esto mismo, aún se puede estudiar la complejidad de problemas de decisión en una cantidad enorme de videojuegos particulares.

Bibliografía

[1] Aloupis, G., Demaine, E. D., Guo, A., and Viglietta, G. (2014). Classic Nintendo games are (computationally) hard. In Ferro, A., Luccio, F., and Widmayer, P., editors, *Fun with Algorithms*, pages 40–51, Cham. Springer International Publishing.

[2] Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press.

[3] Bandipedia (2020). Crash bandicoot (video game). `https://crashbandicoot.fandom.com/wiki/Crash_Bandicoot_(video_game)`. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[4] Bosboom, J., Demaine, E. D., Hesterberg, A., Lynch, J., and Waingarten, E. (2016). Mario kart is hard. In Akiyama, J., Ito, H., Sakai, T., and Uno, Y., editors, *Discrete and Computational Geometry and Graphs*, pages 49–59, Cham. Springer International Publishing.

[5] Breukelaar, R., Demaine, E. D., Hohenberger, S., Hoogeboom, H. J., Kosters, W. A., and Liben-Nowell, D. (2004). Tetris is hard, even to approximate. *International Journal of Computational Geometry & Applications*, 14(01n02):41–68.

[6] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA. Association for Computing Machinery.

[7] Demaine, E. D., Demaine, M. L., and O'Rourke, J. (2000). Pushpush and push-1 are NP-hard in 2D. In *Proceedings of the 12th Canadian Conference on Computational Geometry, Fredericton, New Brunswick, Canada, August 16-19, 2000*.

[8] Demaine, E. D., Lockhart, J., and Lynch, J. (2018). The computational complexity of portal and other 3d video games. In Ito, H., Leonardi, S., Pagli, L., and Prencipe, G., editors, *9th International Conference on Fun with Algorithms, FUN 2018, June 13-15, 2018, La Maddalena, Italy*,

volume 100 of *LIPIcs*, pages 19:1–19:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[9] Demaine, E. D., Viglietta, G., and Williams, A. (2016). Super Mario bros. is harder/easier than we thought. In Demaine, E. D. and Grandoni, F., editors, *FUN*, volume 49 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[10] DonkeyKongWiki (2020). Donkey kong country. `https://donkeykong.fandom.com/wiki/Donkey_Kong_Country`. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[11] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition.

[12] Hearn, R. A. and Demaine, E. D. (2009). *Games, Puzzles, and Computation*. A. K. Peters, Ltd., USA.

[13] Hopcroft, J., Motwani, R., Ullman, J., and tr, A. (2002). *Introducción a la teoría de autómatas, lenguajes y computación*. Pearson Educación.

[14] Itai, A., Papadimitriou, C., and Szwarcfiter, J. (1982). Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686.

[15] Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R. and Thatcher, J., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

[16] KlonoaWiki (2020). Klonoa: Empire of Dreams. `https://klonoa.fandom.com/wiki/Klonoa:_Empire_of_Dreams`. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[17] LooneyTunesWiki (2020). Sheep Raider). `https://looneytunes.fandom.com/wiki/Sheep_Raider`. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[18] MetroidWiki (2020). Metroid (game). `http://www.metroidwiki.org/wiki/Metroid(game)`. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[19] PortalUnofficialWiki (2020). Portal. https://theportalwiki.com/wiki/Portal. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[20] SuperMarioWiki (2020a). Mario kart 64. https://www.mariowiki.com/Mario_Kart_64. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[21] SuperMarioWiki (2020b). Super mario bros. http://www.mariowiki.com/SuperMarioBros. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[22] tetris.com (2020). Tetris. https://tetris.com. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[23] TheBindingOfIsaacRebirthWiki (2020). Binding of Isaac: Rebirth Wiki). https://bindingofisaacrebirth.gamepedia.com/Binding_of_Isaac:_Rebirth_Wiki. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[24] ThePokémonWiki (2020). Pokémon red and blue versions. https://pokemon.fandom.com/wiki/Pok%C3%A9mon_Red_and_Blue_Version. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[25] TheSpritersResource (2020). The Spriters Resource. https://www.spriters-resource.com/. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[26] TheTexturesResource (2020). Crates. https://www.textures-resource.com/playstation/ps1crash1/texture/2960/. [En funcionamiento al momento de la consulta; visitado en Abril-2020].

[27] Viglietta, G. (2014). Gaming is a hard job, but someone has to do it! *Theory Comput. Syst.*, 54(4):595–621.

[28] ZeldaWiki (2020). The legend of zelda: A link to the past. https://zelda.gamepedia.com/The_Legend_of_Zelda:_A_Link_to_the_Past. [En funcionamiento al momento de la consulta; visitado en Abril-2020].