Resolución de puzles Hitori con CLIPS:

Descripción de las reglas y técnicas utilizadas

Javier Ivar Advani Aguilar

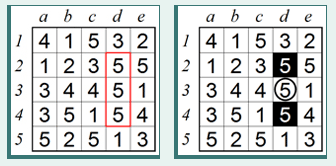
13 de febrero de 2020

**I. Introducción**

La propuesta de resolución presentada, **resuelve en total 36/50 puzles correctamente** (pueden verse pantallazos de cada uno de los juegos que finaliza) inicialmente a través de **52 reglas** (téngase en cuenta que en realidad varios grupos de reglas, algunos de hasta 10 o 12, corresponden a una única estrategia para resolver Hitoris). Si se ha encontrado alguna opción que optimice y reduzca la cantidad de reglas, es indicado como sugerencia, al final de la explicación de la misma. También se incluye enlace a la técnica descrita

**II. Descripción de técnicas generales.**

1. Tres valores repetidos y adyacentes ([adjacent triplets](https://www.conceptispuzzles.com/index.aspx?uri=puzzle/hitori/techniques)).



La celda central debe ser siempre asignada. Las otras dos son eliminadas para evitar romper una de las principales normas del juego: El valor no puede asignarse más de una vez para la fila y columna correspondiente a esa celda. Esta técnica se ha implementado a través de dos reglas, una para filas y otra para columnas. Se muestra a continuación la implementación para columnas, pero pueden verse ambas en el código.

(defrule three-values-repeated-in-column

(declare (salience 3))

?h1<-(celda (fila ?f1) (columna ?c) (valor ?v) (estado desconocido))

?h2<-(celda (fila ?f2) (columna ?c) (valor ?v) (estado desconocido))

?h3<-(celda (fila ?f3) (columna ?c) (valor ?v) (estado desconocido))

(test ( and (eq (- ?f1 ?f2) 1) (eq (- ?f2 ?f3) 1)))

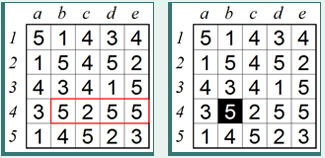
=>

(modify ?h1 (estado eliminado))

(modify ?h2 (estado asignado))

(modify ?h3 (estado eliminado)))

1. Pair induction (fuente [aquí](https://www.conceptispuzzles.com/index.aspx?uri=puzzle/hitori/techniques)).



Dadas 3 celdas de la misma fila o columna con un mismo valor v, siendo 2 de ellas adyacentes entre sí, debemos eliminar la celda apartada. Se ha implementado con 2 reglas (una para filas y otra para columnas). Esta regla no interfiere con la anterior en caso de tener un “triplete”, ya que la anterior se limitaría a asignar a la celda central, y la aquí descrita eliminaría una de las otras dos (aunque veremos más adelante que no es el caso, ya que la implementación de una de las reglas principales va a tener más prioridad).

(defrule pair-induction-row

(declare (salience 3))

(celda (fila ?f) (columna ?c1) (valor ?v))

(celda (fila ?f) (columna ?c2) (valor ?v))

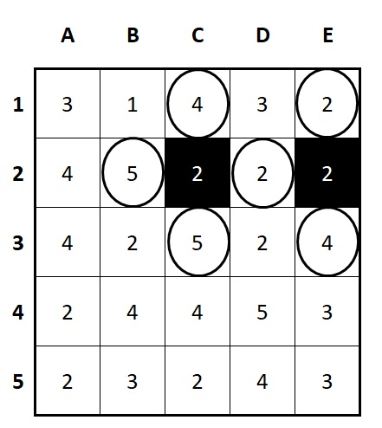
?h<-(celda (fila ?f) (columna ?c3) (valor ?v) (estado desconocido))

(test (and (eq (abs(- ?c1 ?c2)) 1) (and (neq ?c3 ?c2) (neq ?c1 ?c3))))

=>

(modify ?h (estado eliminado)))

1. Asignar alrededor de una celda eliminada.



Como bien se indica en la sección de esta web [CIRCLES AROUND BLACK CELLS](http://tectonicpuzzel.eu/hitori-solving-techniques.html), una de las principales premisas es que una celda negra/eliminada debe estar siempre rodeada de celdas blancas. Otro enfoque para explicar lo mismo [aquí](https://www.conceptispuzzles.com/index.aspx?uri=puzzle/hitori/rules): dos celdas negras nunca pueden ser adyacentes. La primera forma de verlo parece más eficiente y sencilla, porque no requiere ir filtrando múltiples celdas negras, comprobar su posición, desasignarlas si estaban adyacentes, etc. En una primera aproximación, se habían elaborado 2 reglas (fila(s) adyacente(s) y columna(s) adyacente(s) a la eliminada), pero ha sido de mayor utilidad para depurar y localizar la activación de algunas técnicas más complejas tenerlo con 4 reglas (asignar celda adyacente arriba, abajo, izquierda y derecha). Las asignación de una de las celdas al lado de la eliminada es tal que así:

(defrule assign-around-deleted-cell-left

(declare (salience 5))

(celda (fila ?f) (columna ?c1) (estado eliminado))

?h1<-(celda (fila ?f) (columna ?c2) (estado desconocido))

(test (eq (- ?c1 ?c2) 1))

=>

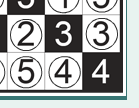
(modify ?h1 (estado asignado))

1. Evitar celdas aisladas.

Tal y como se afirma [aquí](https://www.conceptispuzzles.com/index.aspx?uri=puzzle/hitori/rules): “Cuando el puzle está completado, las celdas no sombreadas (blancas, asignadas) comprenden un área única y continua”. Esto implica, que ninguna celda blanca puede estar completamente aislada. Esta regla, nuevamente, se ha planteado desde un punto de vista disgregado, para poder localizar de manera más efectiva posibles errores al implementar otras. Una celda asignada puede estar en el interior del puzle, en un borde o en una esquina. Para el primera caso, si 3 de las cuatro celdas adyacentes en fila y columna a la asignada son eliminadas, y la cuarta es desconocida, ésta debe ser asignada para garantizar el área continua. Lo mismo si tenemos 2 de 3 adyacentes eliminadas en un borde, y 1 de 2 eliminadas en una esquina.



El valor cuatro que hay asignado en el centro de la imagen posee 3 celdas eliminadas adyacentes, por lo que el valor tres que tiene a su izquierda debe ser asignado, evitando así un aislamiento.



En este caso, al borde del tablero, tenemos un 4 asignado, y 2 celdas adyacentes de las 3 eliminadas. El 5 debe ser asignado.

Comprobar cada uno de los posibles casos de aislamiento da lugar a 4 reglas para una celda interna, 3 para cada borde (un total de 12), y otras dos reglas para verificar cada esquina (8 reglas). Esta es la técnica que más reglas individuales ocupa, con un total de 24. No se le asigna prioridad alguna frente a otras para evitar una comprobación exhaustiva de todas las celdas durante la ejecución.   
  
Dejamos un ejemplo de una de las celdas internas.

(defrule avoid-isolated-cell-margin-down-1

(celda (fila ?f1) (columna ?c2) (estado eliminado))

(celda (fila ?f2) (columna ?c1) (estado eliminado))

(celda (fila ?f2) (columna ?c2) (estado asignado))

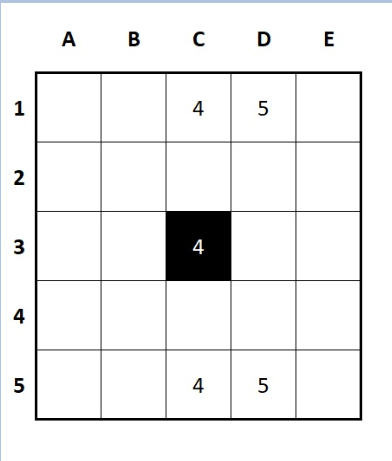
?h1<-(celda (fila ?f2) (columna ?c3) (estado desconocido))

(test (and (and (= ?f1 8)(= ?f2 9)) (and (eq (- ?c2 ?c3) 1) (eq (- ?c1 ?c2) 1))))

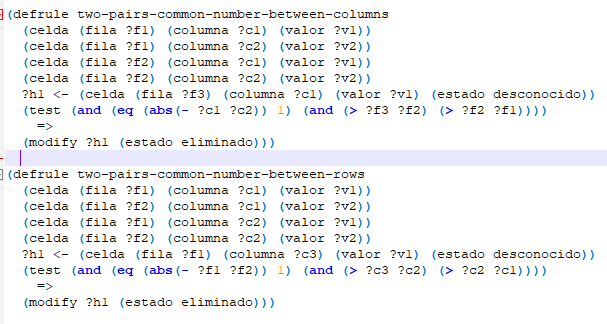
=>

(modify ?h1 (estado asignado)))

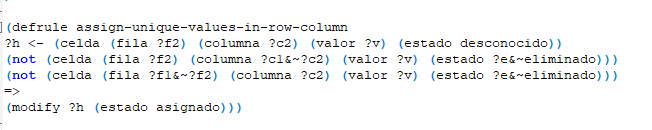
1. Dos parejas con número común en el centro.



Siguiendo la técnica [aquí](http://tectonicpuzzel.eu/hitori-solving-techniques.html) descrita, si tenemos dos parejas de números, con un número entre ellas igual al de la sección correspondiente, la celda común en el centro debe ser eliminada. Se ha implementado una versión para filas y otra para columnas. Se dispara pocas veces, pero ha saltado en algún puzle.

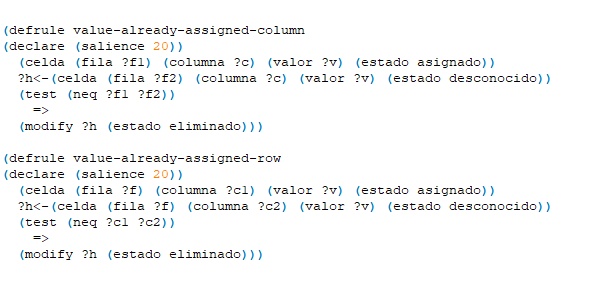


1. Asignar valor único de fila y columna.

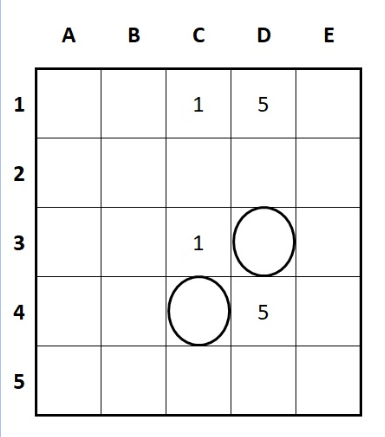
Siguiendo la [primera premisa de Hitori](https://www.conceptispuzzles.com/index.aspx?uri=puzzle/hitori/rules), si el valor de una celda no se repite en toda la fila ni en toda la columna, esa celda debe ser asignada (no es algo que venga directamente indicado, pero se deduce a partir de que ningún número aparece en una fila o columna más de una vez). Para mostrar un uso más diverso de la sintaxis en CLIPS, se ha hecho uso del “not”, para indicar que “no hay ninguna otra celda en toda la fila ni en toda la columna con el mismo valor, por lo que puedo asignarla”.

1. Eliminar valor ya asignado.

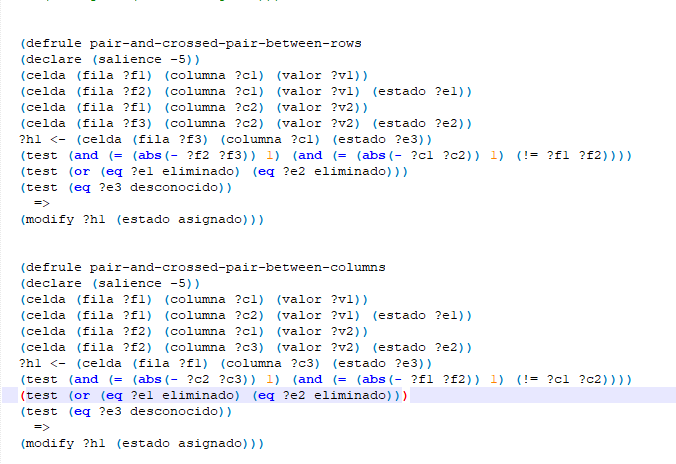
Si ese valor ya ha sido asignado en la misma fila, podemos eliminarlo. Lo mismo para las columnas. Esta regla tiene prioridad 20, porque queremos evitar comprobaciones excesivas si ya hemos asignado un valor con cualquiera de las otras reglas.



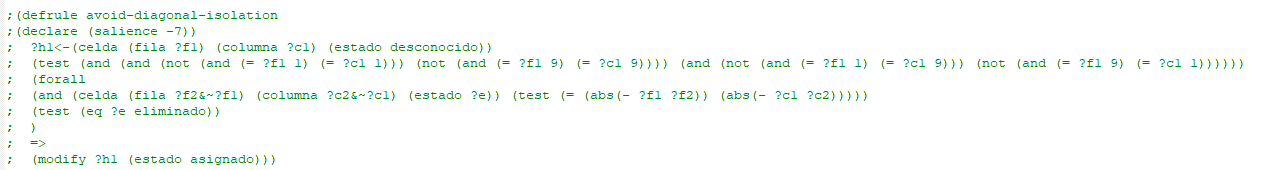
1. Pareja y pareja cruzada.



La diagonal inversa va a ser siempre asignada, si uno de los dos números de la pareja cruzada es eliminado. Se le da poca prioridad, porque ya se tiene una regla que siempre asigna alrededor de cualquiera eliminada.

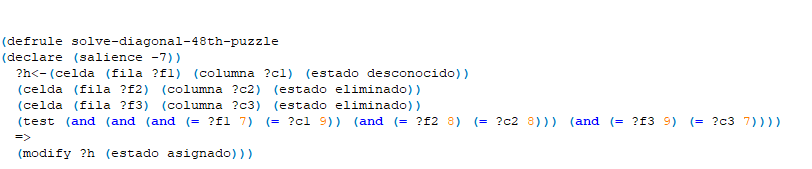


Hasta aquí, con estas reglas descritas, se han podido resolver unos 25 puzles fluidamente.   
  
Una de las técnicas más evidentes que podían aplicarse, observando las celdas no asignadas restantes en los puzles inacabados, era evitar una diagonal que corte las celdas asignadas en dos (el área asignada debe ser una sola y continua). La idea de hacer una única regla para ello, después de numerosos intentos quedó así:

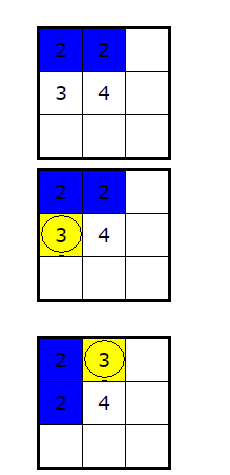


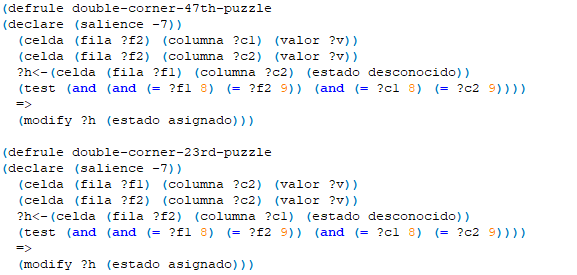
Esta regla sin embargo es incorrecta, porque se dispara, entre otros casos, para diagonales de una única celda, cosa que es incorrecta y potencialmente dañaba los resultados correctos anteriores. Como eran relativamente pocas las diagonales a la vista, se han ido creando reglas específicas para cada uno de estos puzles.

Una simple diagonal especificada en regla, resolvía el resto del tablero sin dificultades, haciendo uso de las otras reglas anteriormente codificadas. Sin embargo, es evidente la ralentización y el coste computacional que tiene cada una de ellas a medida que se fueron implementando, pues estamos forzando comprobaciones en una posición fija del puzle. Se ha dejado con ello un total de 9 reglas de diagonal, en un formato en que se especifica el número de puzle al que corresponde, y las posiciones de celda en que deben comprobarse los estados. Es totalmente obvio que no es la manera más óptima de resolver una situación así, pero era la única aproximación fiable después de numerosos intentos haciendo uso de diferentes versiones del forall anteriormente expuesto.



Las dos últimas instrucciones que se han “forzado” para puzles concretos, corresponden a una regla extraída [aquí](http://www.menneske.no/hitori/methods/eng/methoddc.html). Si tenemos una pareja de celdas adyacentes con el mismo valor haciendo esquina, la celda contigua a la pareja (según que esquina) se asigna. No hemos encontrado una forma genérica de delimitar esta pareja, y tampoco merecía la pena hacerlo para los casos restantes en que teníamos parejas haciendo esquina (se han detectado 2), por lo que hemos codificado esas reglas directamente.





**III. Conclusiones**

Aunque las 11 últimas reglas descritas ralentizan la ejecución en CLIPS considerablemente (actualmente, tarda aproximadamente 1 ó 2 segundos por puzle), se ha considerado que era interesante su implementación, al poder completar con ellas otros 10 u 11 puzles más (de 25 a 36). Unas posibles líneas de mejora, serían indagar algo más en la regla de la diagonal, evidentemente, y tratar de unificar las reglas que evitan el aislamiento de las celdas asignadas en una o dos únicas (por ejemplo haciendo uso de forall) aunque el rendimiento global de la aplicación, sería más o menos parecido (lo único en lo que se gana es en limpieza, porque también reglas muy genéricas, se considera pueden alejarnos de la legibilidad y claridad del código, siguiendo el principio de programación *divide et vinces* ).

**Anexo: Puzles resueltos.**

Puzles 1-26, 28-30, 42-48

