Departamentos de Ciencias e Ingeniería de la Computación.
Lógica para Ciencias de la Computación.
Universidad Nacional del Sur.

Lógica en SWI-Prolog

Para nuestra implementación de resolverGrilla/6, implementamos distintos predicados auxiliares, a continuación detallaremos tanto en ellos como en el resolverGrilla/6.

longitud/2

Determina la longitud de una lista dada.

traspuesta/3

Genera una grilla traspuesta a una grilla dada.

resolverLinea/3

Intenta generar la solución de una línea, intersectando una lista de posibilidades

resolverFilas/4

Completa la mayor cantidad de celdas posibles de una grilla dada, haciendo uso de resolverLinea/3 sobre cada una de las filas.

verificarFilas/4

Verifica que una grilla respete las pistas en cada una de sus filas.

generarPosibilidad/2

Dada una fila que usaremos de base y su pista correspondiente generar las distintas soluciones posibles a esa fila. Esto es posible dado que Prolog puede generar múltiples soluciones a un mismo predicado. Los predicados auxiliares son utilizados para establecer estas combinaciones agregando pistas y espacios.

generarListaPosibilidades/3

Genera una lista de posibilidades haciendo uso de generarPosibilidad/2.

resolverGrilla/6

Es el predicado principal del proyecto y sigue la siguiente estrategia.
Rellena la mayor cantidad de celdas posibles con resolverFilas/4, luego genera la grilla traspuesta a través de traspuesta/3 y vuelve a rellenar celdas. Esto nos permite resolver las pistas de columnas y filas con mismo mecanismo, luego volvemos a generar la grilla traspuesta y en caso de que no se verifiquen las pistas realiza un llamado recursivo, caso contrario retorna la grilla resultante.
Observación: Solo es necesario que se cumplan las pistas de las filas ya que si el tablero las cumple necesariamente está completo.

Departamentos de Ciencias e Ingeniería de la Computación.
Lógica para Ciencias de la Computación.
Universidad Nacional del Sur.

Interfaz en React

Valores mantenidos

En esta nueva versión del juego algunos de los valores almacenados en el estado se modifican y además se agregan nuevos.

Añadimos una nueva grilla, en este caso `grillaResuelta` que almacena la solución del juego para posteriormente ser consultada o brindar pistas al usuario.

Se agrega `mostrandoSolucion` que nos indica si el juego se encuentra en curso (esto es si el usuario está jugando, su valor será false) o si se está mostrando la solución (true).

También se remueve el estado painting para sumar uno nuevo, `seleccion`, que es utilizado de guía para lo que será la próxima jugada. Si el valor es '#' está pintando, si es 'X' está desmarcando, si es 'S' mostrando solución y si es 'P' mostrará pista.

Consultas al servidor y actualización de estado

Al igual que los valores, esta nueva versión del juego añade nuevas consultas y actualizaciones.
Como podemos ver a continuación, a lo anteriormente hecho se le suma una nueva consulta a Prolog, `resolverGrilla/6`, quien nos devuelve (y almacena en grillaResuelta) una grilla con el juego resuelto para posteriormente ser utilizada en pistas o revelar solución.

```
handlePengineCreate() {
  const queryS = 'init(PistasFilas, PistasColumnas, Grilla)';
  this.pengine.query(queryS, (success, response) => {
    if (success) {
      this.setState({
        grid: response['Grilla'],
        pistasEnFilas: response['PistasFilas'],
        pistasEnColumnas: response['PistasColumnas'],
        listaFilas: [].constructor(response['PistasFilas'].length),
        listaColumnas: [].constructor(response['PistasColumnas'].length),
      });
      const querySS = 'resolverGrilla('+JSON.stringify(this.state.pistasEnFilas)+','
        +JSON.stringify(this.state.pistasEnColumnas)+','
        +JSON.stringify(this.state.grid).replaceAll('"_"', "_")+','
        +'GrillaResueltaAux,'
        +(this.state.listaFilas.length-1)+','
        +(this.state.listaColumnas.length-1)+')';
      this.pengine.query(querySS, (success, response) => {
      if(success){
        this.setState({
          grillaResuelta: response['GrillaResueltaAux'],
        })
      }
    })
    }
  });
}
```

El handleClick se modifica, verificando inicialmente cuál es la selección actual y actuando en base a ese valor. Esto es, si el valor de 'seleccion' es '#' o 'X' se procederá de la misma forma que antes. También ocurrirá una inserción si el valor es 'P' y la celda que estamos clickeando está vacía, de lo contrario no se realizará ninguna acción.

```
handleClick(i, j) {
  // No action on click if we are waiting.
  const selActual = this.state.seleccion;
  if (this.state.waiting || selActual === 'S') {
    return;
  }
  if(selActual === 'P' && this.state.grid[i][j] !== "_") {
    return;
  }
  let marca = selActual === "P" ? '""+this.state.grillaResuelta[i][j]+'"' : '"'+selActual+'"';
  console.log("marca",this.state.marcado);
  // Build Prolog query to make the move, which will look as follows:
  // put("#",[0,1],[], [],[["X",_,_,_,_],["X",_,"X",_,_],["X",_,_,_,_],["#","#","#",_,_],[_,_,"#","#","#"]], GrillaRes, FilaSat, ColSat)
  const squaresS = JSON.stringify(this.state.grid).replaceAll('"_"', "_"); // Remove quotes for variables.
  const queryS = 'put('
    + marca
    + ',[' + i + ',' + j + '],'
    + JSON.stringify(this.state.pistasEnFilas) + ','
    + JSON.stringify(this.state.pistasEnColumnas) + ','
    + squaresS
    + ',GrillaRes,FilaSat,ColSat)';
  this.setState({
    waiting: true
  });
```

La función render también sufre modificaciones, añadiendo constantes que posteriormente se utilizarán para pintar (o no) los botones de las opciones, y de esta forma identificar fácilmente el que se encuentra seleccionado.

```
render() {
  if (this.state.grid === null) {
    return null;
  }
  const marcado = this.state.seleccion;
  const pintar = marcado === '#' ? 'optionPaint' : 'optionPaint deshabilitado';
  const marcar = marcado === 'X' ? 'optionMark' : 'optionMark deshabilitado';
  const pista = marcado === 'P' ? 'opcionPista' : 'opcionPista deshabilitado';
  const solucion = marcado === 'S' ? 'opcionSolucion' : 'opcionSolucion deshabilitado';
```

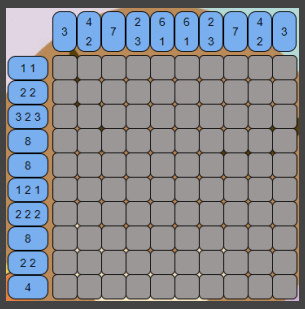
Otra modificación importante ocurre en el componente 'board', donde se agrega la constante grillaVisible, que es quien define cuál será el tablero visible durante el juego. Es decir, mientras el juego esté en curso el usuario podrá interactuar libremente con él, pintando, desmarcando, o consultando pistas. Cuando pida ver la solución, automáticamente el juego se pondrá en pausa y mostrará la grilla resuelta, hasta que el jugador desee continuar con otra opción.

```
class Board extends React.Component {
    render() {
        const { estadoDelJuego, listaFilas, listaColumnas } = this.props;
        const grillaVisible = (estadoDelJuego !== 'Juego en pausa') ? this.props.grid : this.props.grillaResuelta;
        const pistasDeFilas = this.props.pistasEnFilas;
        const pistasDeColumnas = this.props.pistasEnColumnas;
        const isDisabled = estadoDelJuego !== "Juego en curso";
```

Departamentos de Ciencias e Ingeniería de la Computación.
Lógica para Ciencias de la Computación.
Universidad Nacional del Sur.

Manual de usuario

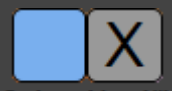
Inicio



Al iniciar el juego se mostrará una grilla con cuadros que deberán ser pintados. Sobre cada fila y cada columna podrán visualizarse pistas, que indican individualmente cuántos cuadros seguidos hay que pintar; a su vez la suma de los números dentro de cada globo indica la cantidad total de cuadros que deben ser pintados en la fila o columna.

Herramientas

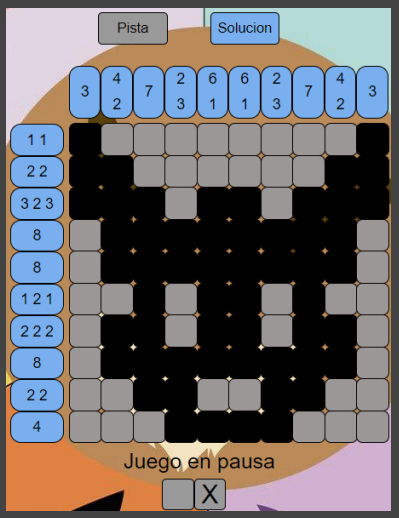
Debajo de la grilla se encuentran ubicados un texto indicando que el juego está en curso y dos botones. El color de estos últimos puede contener celeste si se encuentra habilitado o gris si no lo está. El izquierdo (habilitado por defecto) indica que estamos pintando, mientras el derecho sirve para marcar las celdas que por algún motivo no queremos pintar. Al presionar alguno de los botones automáticamente cambiará el modo de selección. Si mientras estamos pintando hacemos 'click' sobre un cuadro pintado, éste se volverá gris nuevamente, así como si clickeamos una 'X' mientras ésta está seleccionada.



Selección: '#'

Selección: 'X'

Además en el panel superior se encuentran los botones 'Pista' y 'Solución'. El color y funcionamiento de ambos es el mismo que los del panel inferior. Mientras se encuentre seleccionado el botón 'Pista' cualquier cuadro vacío (esto es, ni pintado, ni marcado con 'X') en el que hagamos 'click' revelará automáticamente lo que debería encontrarse debajo, ya sea pintando o marcando con 'X'. El botón solución funciona algo diferente, ya que si lo seleccionamos inhabilitará los cuadros de la grilla y nos mostrará la solución del juego. Mientras este último esté seleccionado sólo se podrán apretar los otros tres botones ('Pista', 'Pintar' o 'X'), de esa forma volveremos a visualizar nuestra grilla y los cuadros de la misma se habilitarán nuevamente.



Selección: 'P'

Avance y finalización del juego

Cuando una fila o columna se completa, es decir se pintan los cuadros indicados en las pistas, el globo de las mismas cambiará de color y se volverá rosa, indicándonos que ya pintamos todos los necesarios. En caso de desmarcar alguno de los cuadros anteriormente pintados, éste volverá a ser azul.
Finalmente, cuando todas las filas y columnas completen sus pistas en simultáneo, se mostrará un alerta al jugador indicando que ganó y se inhabilitarán los botones de la grilla.

Extras:

- Los botones superiores, al igual que los inferiores presentados anteriormente cambian de color al pasar el mouse por encima de los mismos.
- Al formar la figura y ganar el juego los cuadros de la grilla así como los botones de opciones se deshabilitan para su modificación.