Nouhaud (Pereira) Mattéo - 22008937
Patry Simon - 22004482
Proust Jules - 22002224
Rullier Mateo - 22001944

Théorie des langages et automates

Projet

Rendu Jalon 1

Ce document était dans une archive .zip, accompagné d'un fichier tableur, ce-dernier contient les annexes nécessaires à la compréhension de ce rendu

Lexique

Notre langage est composé du lexique suivant :

,	EXCEPT	LOOP	SPAWN
;	EXIT	ON	SWITCH
(	FROM	OPEN	TO
)	GHOST	OR	TP
ALL	GOING	PLACE	U | UP
AND	GOTO	PLAYER	WALL
CLOSE	IS_OFF	R | RIGHT	WHEN
CREATE	IS_ON	REPEAT	X
D | DOWN	L | LEFT	SET	Y
DOOR	LEVEL	SIZE	num

L'analyse lexicale est disponible dans la première feuille du tableur *projet.xlsx* ci-joint.

Notes :

- Les mots RIGHT et R (tout comme L et LEFT, …) renvoient vers le même état final, car une fois dans le langage, utilisé l'un ou l'autre ne fait aucune différence.
- num correspond à un entier positif.
- IS ON et IS OFF sont considérés comme des mots à part entière, IS n'est pas un mot (ON et OFF non plus).
- Le code peut accueillir du commentaire, matérialisé par ce qu'on veut précéder d'un double-slash (//) en fin de ligne.
- Pour tous les mots du langage, l'écriture en majuscule ou minuscule n'a pas d'importance.

Grammaire et table d'analyse

Les 2e et 3e feuilles du tableur *projet.xlsx*, ci-joint, contiennent l'analyse syntaxique et sa table.

Ci-dessous, la grammaire de notre langage, les différents symboles non-terminaux sont matérialisés ici en bleu (pour le lien avec le tableur, les nombres font foi), et les symboles terminaux en orange:

```
// Base
    000S → CREATE LEVEL SIZE 900position ; 001player 002place

    001player → *
    001player → SET PLAYER ON 900position ;
```

```
    002place → *
    002place → PLACE 005entity ; 002place

    005entity → WALL ON 100positionGroup 105moreLessPosition
    005entity → EXIT ON 100positionGroup 105moreLessPosition
    005entity → TP 030tp
    005entity → GHOST 040ghost
    005entity → SWITCH num 050switch
    005entity → DOOR 060door

    // Différents ordonnancement des parties
        030tp → ON 100positionGroup 105moreLessPosition 031
        030tp → FROM 920rightLeftDownUp 032
        030tp → GOTO 900position 033

        031 → FROM 920rightLeftDownUp GOTO 900position
        031 → GOTO 900position FROM 920rightLeftDownUp

        032 → ON 100positionGroup 105moreLessPosition GOTO 900position
        032 → GOTO 900position ON 100positionGroup 105moreLessPosition

        033 → FROM 920rightLeftDownUp ON 100positionGroup 105moreLessPosition
        033 → ON 100positionGroup 105moreLessPosition FROM 920rightLeftDownUp

        040ghost → 401 041
        040ghost → SPAWN ON 900position 042
        040ghost → 450 043

        041 → SPAWN ON 900position 450
        041 → 450 SPAWN ON 900position

        042 → 401 450
        042 → 450 401

        043 → SPAWN ON 900position 401
        043 → 401 SPAWN ON 900position

        050switch → ON 900position 501
        050switch → SET 910isOnOff ON 900position

        060door → ON 100positionGroup 105moreLessPosition 601
        060door → 602 ON 100positionGroup 105moreLessPosition

// Général
    900position → X 901
    900position → Y 902
```

```
    901 → num Y num
    901 → Y ( num num )

    902 → num X num
    902 → X ( num num )

    910isOnOff → IS_ON
    910isOnOff → IS_OFF

    920rightLeftDownUp → R|RIGHT
    920rightLeftDownUp → L|LEFT
    920rightLeftDownUp → D|DOWN
    920rightLeftDownUp → U|UP

    930orAnd → AND
    930orAnd → OR

// Partie ON ...
    100positionGroup → ALL
    100positionGroup → X 101
    100positionGroup → Y 102

    101 → 120 140
    101 → Y ( num num 131)

    102 → 120 141
    102 → X ( num num 131)

    105moreLessPosition → *
    105moreLessPosition → ON 100positionGroup 105moreLessPosition
    105moreLessPosition → EXCEPT 100positionGroup 105moreLessPosition

    120 → num
    120 → ( num 121)

    121 → *
    121 → , num 121
    121 → TO num 122

    122 → *
    122 → , num 121

    131 → *
    131 → , num num 131

    140 → *
```

```
    140 → Y 120

    141 → *
    141 → X 120

// Partie GHOST
    401 → GOING 402
    401 → GOTO 900position

    402 → 404
    402 → ( 404 403 )

    403 → *
    403 → , 404 403

    404 → 920rightLeftDownUp
    404 → REPEAT 920rightLeftDownUp num

    450 → *
    450 → LOOP

// Partie SWITCH
    501 → *
    501 → SET 910isOnOff

// Partie DOOR
    601 → *
    601 → 602

    602 → OPEN WHEN 603
    602 → CLOSE WHEN 603

    603 → ( 605 ) 910isOnOff 604
    603 → SWITCH num 910isOnOff 604
    603 → 910isOnOff 607 604

    604 → *
    604 → 930orAnd 603

    605 → SWITCH num 606
    605 → ( 605 ) 606

    606 → *
    606 → 930orAnd 605

    607 → ( 605 )
    607 → SWITCH num
```

La grammaire peut aussi être caractérisée de manière plus lisible :

```
{ ?1 | ?2 } = ?1 OU ?2
{ ?1 <> ?2 } = ?1 ET ?2 , peu importe l'ordre
[ ? ] = ? pas obligatoire
[? ...] = se répète autant de fois que nécessaire, avec un ? entre chaque occurrence

CREATE LEVEL SIZE coordSimple ;
[ SET PLAYER ON coordSimple ; ]
{ PLACE { WALL | EXIT } ON coordCompl ;
| PLACE TP { ON coordCompl <> FROM side <> GOTO coordSimple } ;
| PLACE GHOST { SPAWN ON coordSimple <> { GOING {side|(side [, ...])} | GOTO
coordSimple } <> [LOOP] } ;
| PLACE SWITCH num { ON coordSimple <> [ SET { IS ON | IS OFF } ] } ;
| PLACE DOOR { ON coordCompl <> [ { OPEN | CLOSE } WHEN cond ] } ;
} [...]

où coordSimple est :
    { { XY | YX } ( num num ) | { Xnum <> Ynum } }

où coordCompl est :
    { ALL | { XY | YX } ( num num [, ...] ) | { X { num | ( num [, ...] ) } <> Y {
num | ( num [, ...] ) } } | { X | Y } { num |( num [, ...] ) } } } [{ON|EXCEPT} ...]

où side est :
    { R | RIGHT | L | LEFT | D | DOWN | U | UP }

où cond est :
    { {IS ON|IS OFF} <> { SWITCH num | ( SWITCH num [{AND|OR} ...] ) } } [{AND|OR}
...]
    La parenthèse de la condition peut être récursive, peut être accepter :
        IS ON (SWITCH 1 AND (SWITCH 2 OR SWITCH 3))
```

Paramétrage

Nous avons ajouté la possibilité à l'utilisateur du programme de définir la taille (nombres de colonnes (X) et de lignes (Y)), cela se caractérise par la première phrase du fichier qui requière d'indiquer des coordonnées.
L'utilisateur peut aussi choisir l'emplacement de départ du joueur en X et Y, s'il ne l'indique pas, le joueur sera sur la première case par défaut, cette indication se fait forcément après la création du niveau, et avant de placer une quelconque autre entité.
Nous avons les entités présentes dans le programme initial :

- *WALL* (mur), qui prend en paramètre une position ou une liste de position (pour en ajouter plusieurs à la fois) (ON …).
- *EXIT* (sortie), qui prend en paramètre une position ou une liste de position (pour en ajouter plusieurs à la fois) (ON …).

- *TP* (téléporteur, trappe), qui prend en paramètre une ou plusieurs position(s) d'entrée (ON …), un sens de provenance du personnage (FROM …), et une position de sortie (GOTO …).
- *GHOST* (fantôme), qui prend en paramètre une position d'apparition du fantôme (SPAWN ON …), ainsi qu'un schéma de directions jusqu'à sa position final (GOING …), ou qu'une position qui devra se situer sur le même axe X ou Y (GOTO …), et l'argument non-obligatoire LOOP, qui par sa présence indique qui le fantôme devra faire le chemin indiqué en sens inverse, sans ça, il se téléportera à sa position d'apparition.
- *SWITCH* (commutateur), qui prend en paramètre un identifiant (num), une position de départ (ON …), et un état initial (SET IS ON ou SET IS OFF), qui, si non-indiqué, prendra la valeur par défaut IS OFF.
- *DOOR* (porte), qui prend en paramètre une ou plusieurs position(s) (ON …), ainsi qu'une condition définissant si la porte est ouverte ou fermé en fonctions de l'état d'un ou plusieurs commutateur(s), si cette condition n'est pas indiquée, la porte restera fermée par défaut.

Si le développement Java est rapide et nous laisse du temps, nous pourrons peut-être ajouter de nouveaux éléments, eux aussi paramétrables.

Exemples

```
// Niveau 1 – Tiré du programme Java fournit pour le projet
CREATE LEVEL SIZE X20 Y14;

PLACE EXIT ON Y9 X9;

PLACE WALL ON XY(6 1, 14 1);
PLACE WALL ON X(1 TO 20) Y2 EXCEPT X (5, 7, 13, 15) Y 2;
PLACE WALL ON X(6, 10, 12, 14, 19) Y3;
PLACE WALL ON Y (4) EXCEPT X (1, 9, 11, 13, 16, 18, 20);
PLACE WALL ON Y 5 ON X (8, 10, 12, 17, 19);
PLACE WALL ON Y 6 EXCEPT X (1, 7, 9, 11, 18, 20);
PLACE WALL ON Y 7 ON X (2, 10, 19);
PLACE WALL ON Y 8 EXCEPT X (1, 3, 13, 20);
PLACE WALL ON Y 9 ON X (2, 4, 12, 14, 16);
PLACE WALL ON Y 10 EXCEPT X (1, 3, 5, 11, 13, 15, 17);
PLACE WALL ON Y 11 ON X (2, 4, 10, 12, 14, 16);
PLACE WALL ON Y 12 EXCEPT X (1, 2, 3, 9, 11, 13, 15, 20);
PLACE WALL ON Y 13 ON X (1, 2, 4, 10, 14);
PLACE WALL ON Y 14 EXCEPT X (1, 2, 3);

PLACE TP ON X 4 Y 1 FROM LEFT GOTO X 11 Y 3;
PLACE TP FROM RIGHT ON X 6 Y 5 GOTO X 9 Y 7;

PLACE GHOST SPAWN ON x9 y5 GOING (D, D, REPEAT LEFT 5);
PLACE GHOST SPAWN ON x1 y12 GOING (R,R,D,D,L,L) LOOP;
PLACE GHOST GOING (REPEAT DOWN 4) SPAWN ON xy(13 8) ;
```

```
PLACE SWITCH 1 ON x4 y3 SET IS OFF;

PLACE DOOR ON x2 y2 OPEN WHEN SWITCH 1 IS ON;

// Niveau fictif
CREATE LEVEL SIZE XY(5 5);

SET PLAYER yX(1 2) ;

PLACE WALL ON x1 y1;
PLACE WALL ON y2;
PLACE WALL ON y4 EXCEPT x(1);

PLACE TP GOTO yx(3 5) FROM L ON y1 EXCEPT x(1 TO 4);

PLACE SWITCH1 ON xy(1 3) SET IS ON;
PLACE SWITCH 2 ON x1 y5;

PLACE DOOR CLOSE WHEN IS ON (SWITCH 1 OR SWITCH2) ON x4y5;

PLACE GHOST GOTO x2y3 ON y3x4;

PLACE EXIT ON Y 5 X5;
```

Explication Position(s)

- La grille de jeu est définie lors du CREATE LEVEL dans le fichier, celui-ci indique le nombre de colonne (X) et de ligne (Y) à créer.
- Par soucis de compréhension, la première colonne du niveau est X 1 (et non X 0), et la première ligne du niveau est Y 1 (et non Y 0).
- Lors de la définition d'une ou de plusieurs position(s) par XY ou YX, la suite de 2 nombres entre les parenthèses suivantes doit être agencée dans l'ordre indiqué (Ex : XY(1 2) = X1 Y2 , YX(1 2, 2 3) = Y1 X2, Y2 X3).
- Le nombre après un X ou Y peut être collé (sans espace) (Ex : X1y2 possible).

Contacts

Nouhaud (Pereira) Mattéo : matteo.nouhaud@etu.univ-tours.fr
Patry Simon : simon.patry@etu.univ-tours.fr
Proust Jules : jules.proust@etu.univ-tours.fr
Rullier Mateo : mateo.rullier@etu.univ-tours.fr