La logique facile avec TouIST

Formalisez et résolvez facilement des problèmes avec des solveurs SAT, SMT ou QBF

Olivier Gasquet[1], Andreas Herzig[2], Dominique Longin[2], Frédéric Maris[1], Maël Valais[1]
IRIT (Institut de Recherche en Informatique de Toulouse)
[2]CNRS, [1]Université Paul Sabatier, Toulouse, France
{Olivier.Gasquet,Andreas.Herzig,Dominique.Longin,Frederic.Maris,Mael.Valais}@irit.fr

TouIST

(Toulouse Integrated Satisfiability Tool)

Un compilateur de langages logiques étendus et de haut niveau vers des prouveurs efficaces et indépendants

```
0_a_perdu <=>
bigor $t in $T when $t>0:
  tour_de_0($t) and reste($t,0)
          and not reste($t-1,1)
end
```

Problème en langage TouIST

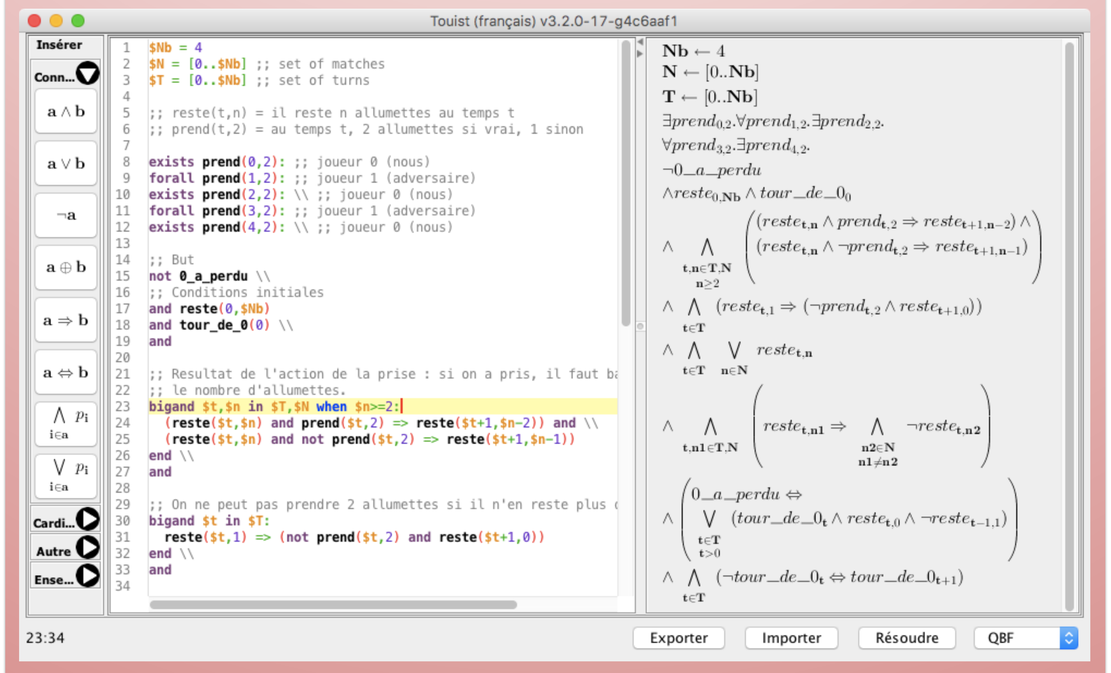
→ Programme TouIST →

```
reste(5,0) -> vrai
0_a_perdu  -> faux
```

Solution

Spécificités

- Langage riche
- Édition visuelle
- Multiples solveurs (SAT, SMT, QBF)
- Facilement extensible
- Projet github, licence MIT



Exemple : le jeu de Nim

- Chacun son tour ($t \in T$), un joueur prend 1 ou 2 allumettes ($n \in A$)
- Celui qui ne peut plus prendre d'allumettes a perdu
- Nous jouons le rôle du **joueur 0** et l'adversaire est le **joueur 1**

Étape 1. Définir les règles du jeu

(Formalisation des règles en logique)

Jouer. Selon son choix, le joueur laisse **une** ou **deux** allumettes en moins :

$$\bigwedge_{\substack{t\in T\\ n\in A\\ n\geq 2}} \begin{pmatrix} (reste(t,n) \wedge \neg prend_2(t) \rightarrow reste(t+1, n-\mathbf{1})) \\ \wedge\, (reste(t,n) \wedge prend_2(t) \rightarrow reste(t+1, n-\mathbf{2})) \end{pmatrix}$$

Dernier coup. S'il ne reste plus qu'**une** allumette, le joueur n'a pas le choix, il doit la prendre et ne laisse **aucune** allumette :

$$\bigwedge_{t\in T} (reste(t,\mathbf{1}) \rightarrow \neg prend_2(t) \wedge reste(t+1,\mathbf{0}))$$

Perdre. Le **joueur 0** a perdu ssi à un tour t, c'est à lui de jouer et il ne reste **aucune** allumette :

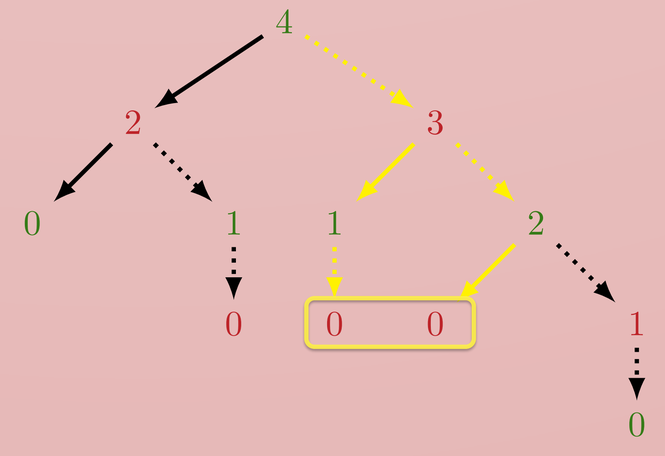
$$\mathbf{0}_perd \leftrightarrow \bigvee_{\substack{t\in T\\ t>0}} (tour_de_\mathbf{0}(t) \wedge reste(t,\mathbf{0}))$$

Étape 2. Jouer et gagner à tous les coups !

(Formalisation d'une stratégie gagnante à l'aide de QBF)

Quels coups le **joueur 0** doit-il faire pour **être sûr de gagner** quels que soient les coups du **joueur 1** ?

$$\begin{bmatrix} \exists prend_2(0). \\ \forall prend_2(1). \\ \exists prend_2(2). \\ \forall prend_2(3). \\ \neg\mathbf{0}_perd \wedge \boldsymbol{\Phi} \end{bmatrix}$$



où $\boldsymbol{\Phi}$ représente les règles du jeu présentées en étape 1

www.irit.fr/touist