AUTOMATED REASONING

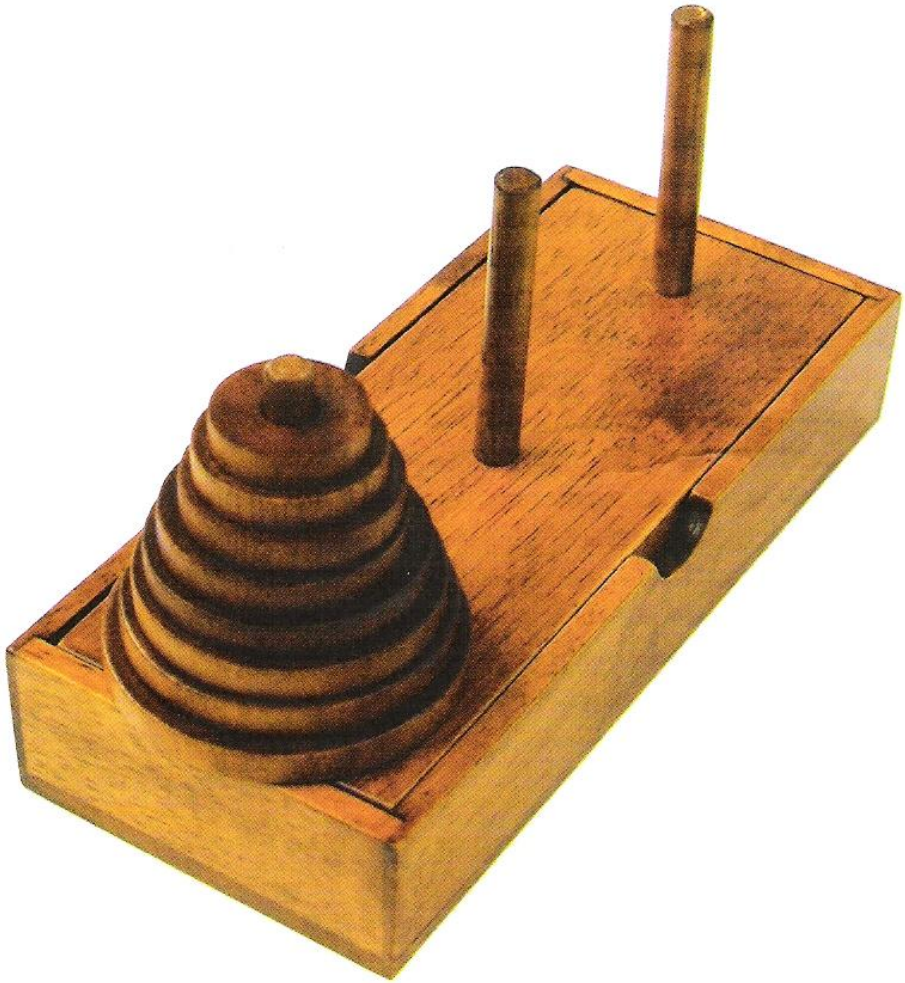
Agostino Dovier

Università di Udine
CLPLAB

Udine, December 2016

The Hanoi Tower

The Hanoi Tower



THE HANOI TOWER

Representation using `inpeg(time, disk, peg)` and `on(time, disk above, disk below/floor)`.
The `top(time, peg, disk)` is defined from the other predicates.

```
peg(1..3).
disk(1..n).
tempo(0..t).

% Initial state (you can use others)
inpeg(0,D,1)   :- disk(D).
on(0,D,D+1)    :- disk(D),disk(D+1).
on(0,n,floor).

% Goal state (all in the 2nd peg)
goal(D) :- inpeg(t,D,2), disk(D).
:- disk(D), not goal(D).
```

THE HANOI TOWER

Representation using `inpeg(time, disk, peg)` and `on(time, disk above, disk below/floor)`.
The `top(time, peg, disk)` is defined from the other predicates.

```
peg(1..3).
disk(1..n).
tempo(0..t).

% Initial state (you can use others)
inpeg(0,D,1)   :- disk(D).
on(0,D,D+1)    :- disk(D),disk(D+1).
on(0,n,floor).

% Goal state (all in the 2nd peg)
goal(D) :- inpeg(t,D,2), disk(D).
:- disk(D), not goal(D).
```

HANOI TOWER

Action: `move(time,pegX, pegY)`

```
% There is one and only one move at time T
1{ move(T,X,Y): peg(X), peg(Y), X != Y} 1 :-
            tempo(T), T < t.

% Smaller above
:- on(T,A,B), tempo(T), disk(A), disk(B), A >= B.

% If there are no disks, no moves
:- move(T,A,B), empty(T,A), tempo(T), peg(A), peg(B).
```

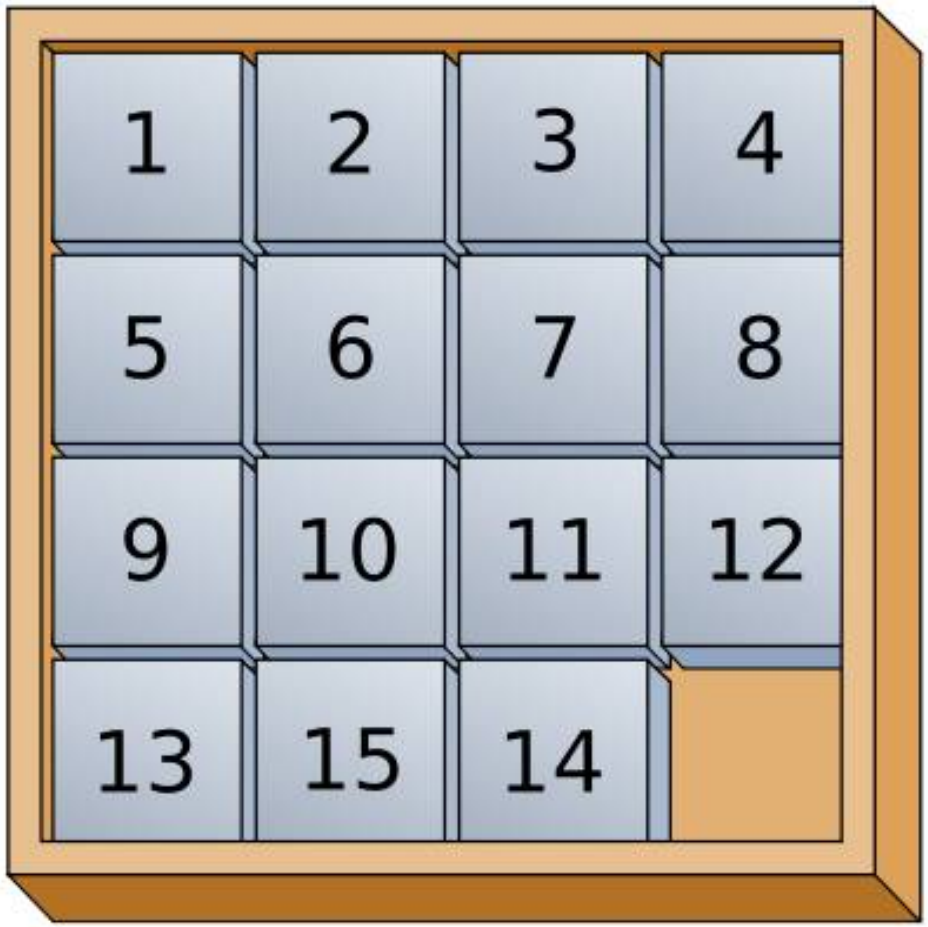
Auxiliary predicates:

```
covered(T,D2)   :- on(T,D1,D2),
                   tempo(T),  disk(D1), disk(D2).

top(T,A,D)      :- inpeg(T,D,A), not covered(T,D),
                   tempo(T), disk(D), peg(A).

nonempty(T,A)   :- inpeg(T,D,A),
                   tempo(T), disk(D), peg(A).

empty(T,A)      :- not nonempty(T,A),
                   tempo(T),  peg(A).

top(T,A,floor) :- empty(T,A),
                   tempo(T), peg(A).

moved(T,D)      :- top(T,A,D), move(T,A,B),
                   tempo(T), disk(D), peg(A), peg(B).
```

HANOI TOWER

```
% 1. change on of moved disk
on(T+1,D1,D2) :-
        top(T,A,D1), top(T,B,D2), move(T,A,B),
        tempo(T), tempo(T+1), peg(A), peg(B), disk(D1), disk(D2).
on(T+1,D,floor) :-
        top(T,A,D), move(T,A,B), empty(T,B),
        tempo(T), tempo(T+1), peg(A), peg(B), disk(D).
% 2. change inpeg of moved disk
inpeg(T+1,D,B) :-
        top(T,A,D), move(T,A,B),
        tempo(T), tempo(T+1), peg(A), peg(B),disk(D).
% Inertia
on(T+1,D,floor) :-
        on(T,D,floor), not moved(T,D),
        tempo(T), tempo(T+1), disk(D).
on(T+1,D1,D2) :-
        on(T,D1,D2), not moved(T,D1),
        tempo(T), tempo(T+1), disk(D1), disk(D2).
inpeg(T+1,D,A) :-
        inpeg(T,D,A), not moved(T,D),
        tempo(T), tempo(T+1), peg(A), disk(D).
```

SAM LLOYD'S PUZZLE



SAM LLOYD'S PUZZLE

```
tempo(0..t).
val(0..8).
range(0..2).

% 0     1     2     %  X=0
% 3     4     5     %  X=1
% 6     7     8     %  X=2
%%%%%%%%
% 0     1     2   <- Y
%%%%%%
%%  "0" is the empty cell (the hole)
%% Define the predicate
%% cell(time T, row X, column Y, value  V)

%%% Input (example)
cell(0,0,0,1). cell(0,0,1,2). cell(0,0,2,5).
cell(0,1,0,3). cell(0,1,1,4). cell(0,1,2,8).
cell(0,2,0,6). cell(0,2,1,0). cell(0,2,2,7).

%%% Goal
goal(X,Y) :- range(X), range(Y), cell(t,X,Y,3*X+Y).
:- range(X),range(Y), not goal(X,Y).
```

SAM LLOYD'S PUZZLE

```
%%% The move moves the hole
1{ move(T,up); move(T,down); move(T,left); move(T,right)}1 :-
        tempo(T), tempo(T+1).

hole(T,X,Y) :- cell(T,X,Y,0), tempo(T), range(X), range(Y).

% Forbidden moves
:- tempo(T), move(T,up),    range(Y), hole(T,0,Y).
:- tempo(T), move(T,down),  range(Y), hole(T,2,Y).
:- tempo(T), move(T,left),  range(X), hole(T,X,0).
:- tempo(T), move(T,right), range(Y), hole(T,X,2).

% New position for the hole
moved(T,X-1,Y) :- hole(T,X,Y), move(T,up),    tempo(T),range(X),range(Y).
moved(T,X+1,Y) :- hole(T,X,Y), move(T,down),  tempo(T),range(X),range(Y).
moved(T,X,Y-1) :- hole(T,X,Y), move(T,left),  tempo(T),range(X),range(Y).
moved(T,X,Y+1) :- hole(T,X,Y), move(T,right), tempo(T),range(X),range(Y).
```

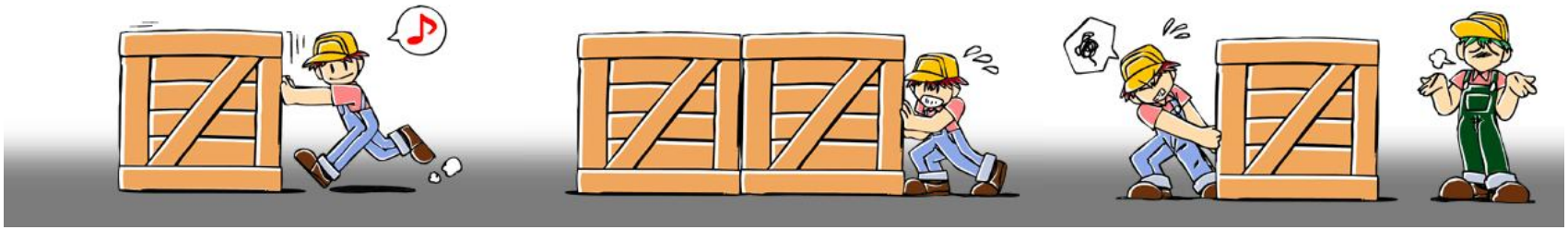
SAM LLOYD'S PUZZLE

```
% New cell contents
cell(T+1,X,Y,0)      :-
     moved(T,X,Y), tempo(T), tempo(T+1), range(X), range(Y).
cell(T+1,X1,Y1,V)    :-
    hole(T,X1,Y1), moved(T,X2,Y2), cell(T,X2,Y2,V),
    tempo(T), tempo(T+1), val(V),
    range(X1), range(X2), range(Y1), range(Y2).

% Inertia
affected(T,X,Y) :- hole(T,X,Y),  tempo(T),  range(X), range(Y).
affected(T,X,Y) :- moved(T,X,Y), tempo(T),  range(X), range(Y).

cell(T+1,X,Y,V) :-  cell(T,X,Y,V),
       not affected(T,X,Y),
       tempo(T), tempo(T+1), range(X), range(Y), val(V).
```

SOKOBAN

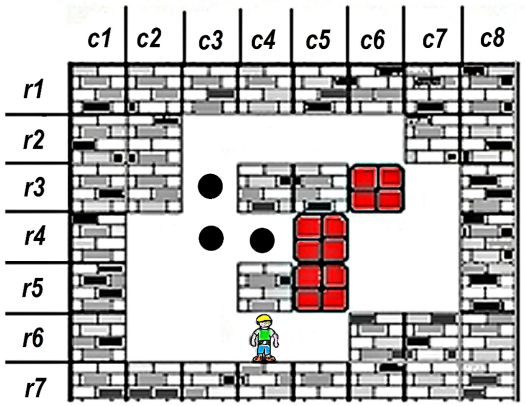
RULES

- Sokoban is a puzzle invented by *Hiroyuki Imabayashi* in 1980
- Sokoban means significa warehouseman (magazziniere) in Japanese
- It is one of the first videogames. It has only three rules.



SOKOBAN

REPRESENTATION

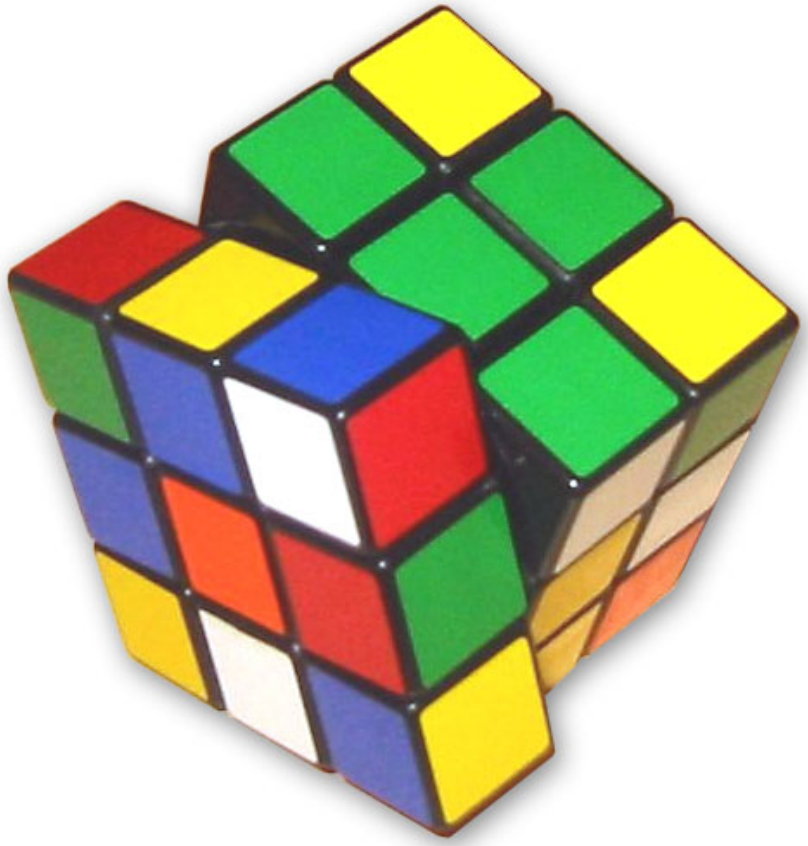



```
top(c2r5,c2r4).          right(c3r2,c4r2).      right(c3r6,c4r6).
top(c2r6,c2r5).          right(c4r2,c5r2).      right(c4r6,c5r6).
top(c3r3,c3r2).          right(c5r2,c6r2).
top(c3r4,c3r3).          right(c6r3,c7r3).      box(c6r3).
top(c3r5,c3r4).          right(c2r4,c3r4).      box(c5r4).
top(c3r6,c3r5).          right(c3r4,c4r4).      box(c5r5).
top(c5r5,c5r4).          right(c4r4,c5r4).
top(c5r6,c5r5).          right(c5r4,c6r4).      storage(c3r3).
top(c6r3,c6r2).          right(c6r4,c7r4).      storage(c3r4).
top(c6r4,c6r3).          right(c2r5,c3r5).      storage(c4r4).
top(c6r5,c6r4).          right(c5r5,c6r5).
top(c7r4,c7r3).          right(c6r5,c7r5).      sokoban(c4r6).
top(c7r5,c7r4).          right(c2r6,c3r6).
```

Solve it defining the predicate `push`(source cell,direction,destination cell), where direction is up, down, left, right.

Peg solitaire

RUBIK'S CUBE



RUBIK'S CUBE

BRAINTWIST

12 COINS

I HAVE NOT A PROGRAM FOR IT!

- There are 12 coins
- Exactly one of them is false
- The false one weights differently from the others (heavier or lighter? Unknown)

- We have a balance:
- How can you discover the false coin in 3 weights?

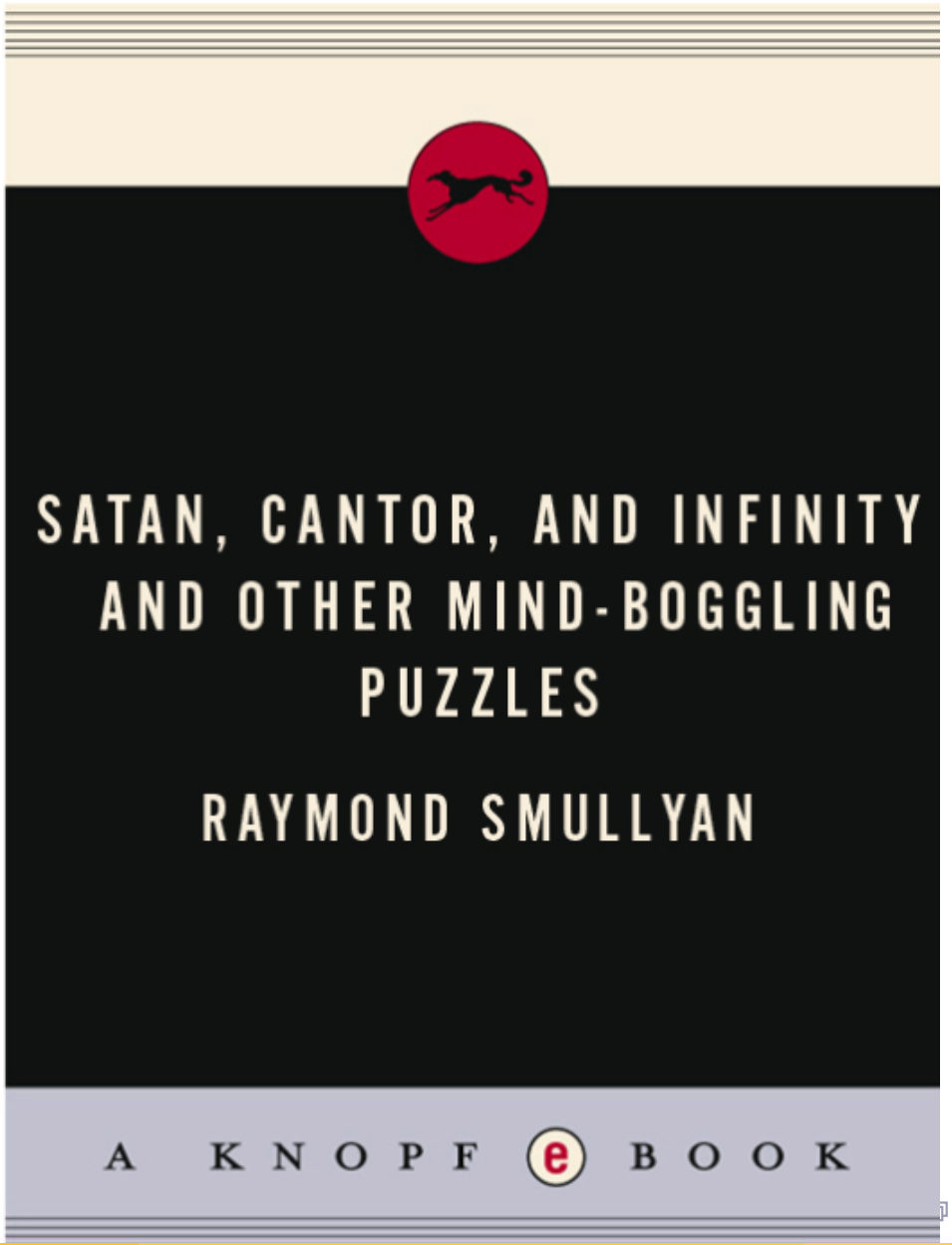
12 COINS

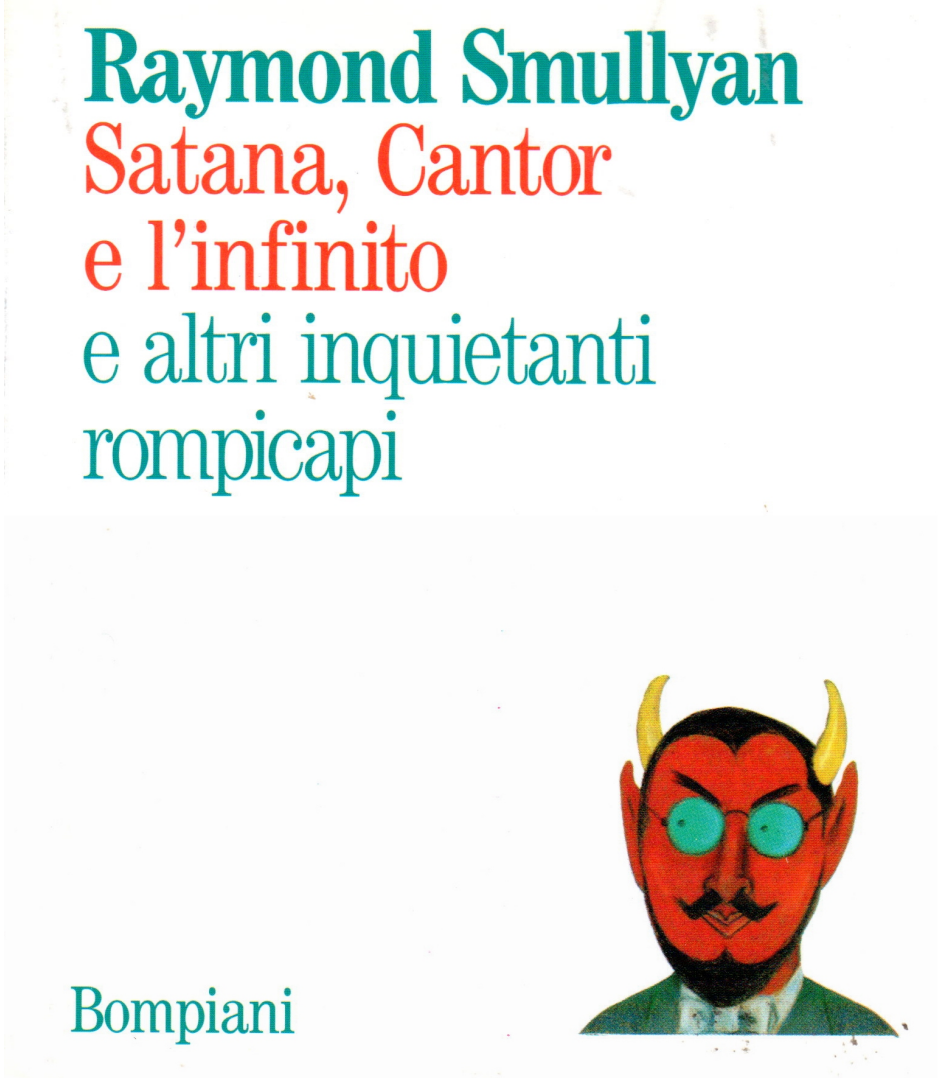
I HAVE NOT A PROGRAM FOR IT!

- There are 12 coins
- Exactly one of them is false
- The false one weights differently from the others (heavier or lighter? Unknown)

- We have a balance:
- How can you discover the false coin in 3 weights?

SMULLYAN


SATAN, CANTOR, AND INFINITY
AND OTHER MIND-BOGGLING
PUZZLES
RAYMOND SMULLYAN
A KNOPF e BOOK


Raymond Smullyan
Satana, Cantor
e l'infinito
e altri inquietanti
rompicapi
Bompiani

1

THE LIE DETECTIVE

WITH A TWINGE of apprehension such as he had never felt before, an anthropologist named Abercrombie stepped onto the Island of Knights and Knaves. He knew that this island was populated by most perplexing people: knights, who make only true statements, and knaves, who make only false ones. "How," Abercrombie wondered, "am I ever to learn anything about this island if I can't tell who is lying and who is telling the truth?"

Abercrombie knew that before he could find out anything he would have to make one friend, someone whom he could always trust to tell him the truth. So when he came upon the first group of natives, three people, presumably named Arthur, Bernard, and Charles, Abercrombie thought to himself, "This is my chance to find a knight for myself." Abercrombie first asked Arthur, "Are Bernard and Charles both knights?" Arthur replied, "Yes." Abercrombie then asked: "Is Bernard a knight?" To his great surprise, Arthur answered: "No." Is Charles a knight or a knave?

Abercrombie was then informed by the one of the three he knew to be a knight that the island had a sorcerer.

"Oh, good!" Abercrombie exclaimed. "We anthropologists are particularly interested in sorcerers, witch doctors, medicine men, shamans, and the like. Where do I find him?"

"You must ask the King," came the reply.

Well, the anthropologist was able to obtain an audience with the King and told him that he wished to meet the Sorcerer.

"Oh, you can't do that," said the King, "unless you first meet his apprentice. If the Sorcerer's Apprentice approves of you, then he will allow you to meet his master; if he doesn't, then he won't."

"The Sorcerer has an apprentice?" asked the anthropologist.

"He certainly does!" replied the King. "There is a famous musical composition about him—I believe the composer was Dukas. Anyway, if you wish to meet the Sorcerer's Apprentice, he is now at his home, which is the third house on Palm Grove. At the moment he is entertaining two guests. If, when you arrive, you can deduce which of the three present is the Sorcerer's Apprentice, I believe that will impress him sufficiently that he will allow you to meet the Sorcerer. Good luck!"

A short walk brought the anthropologist to the house. When he entered, there were indeed three people present.

"Which of you is the Sorcerer's Apprentice?" asked Abercrombie.

"I am," replied one.

"I am the Sorcerer's Apprentice!" cried a second.

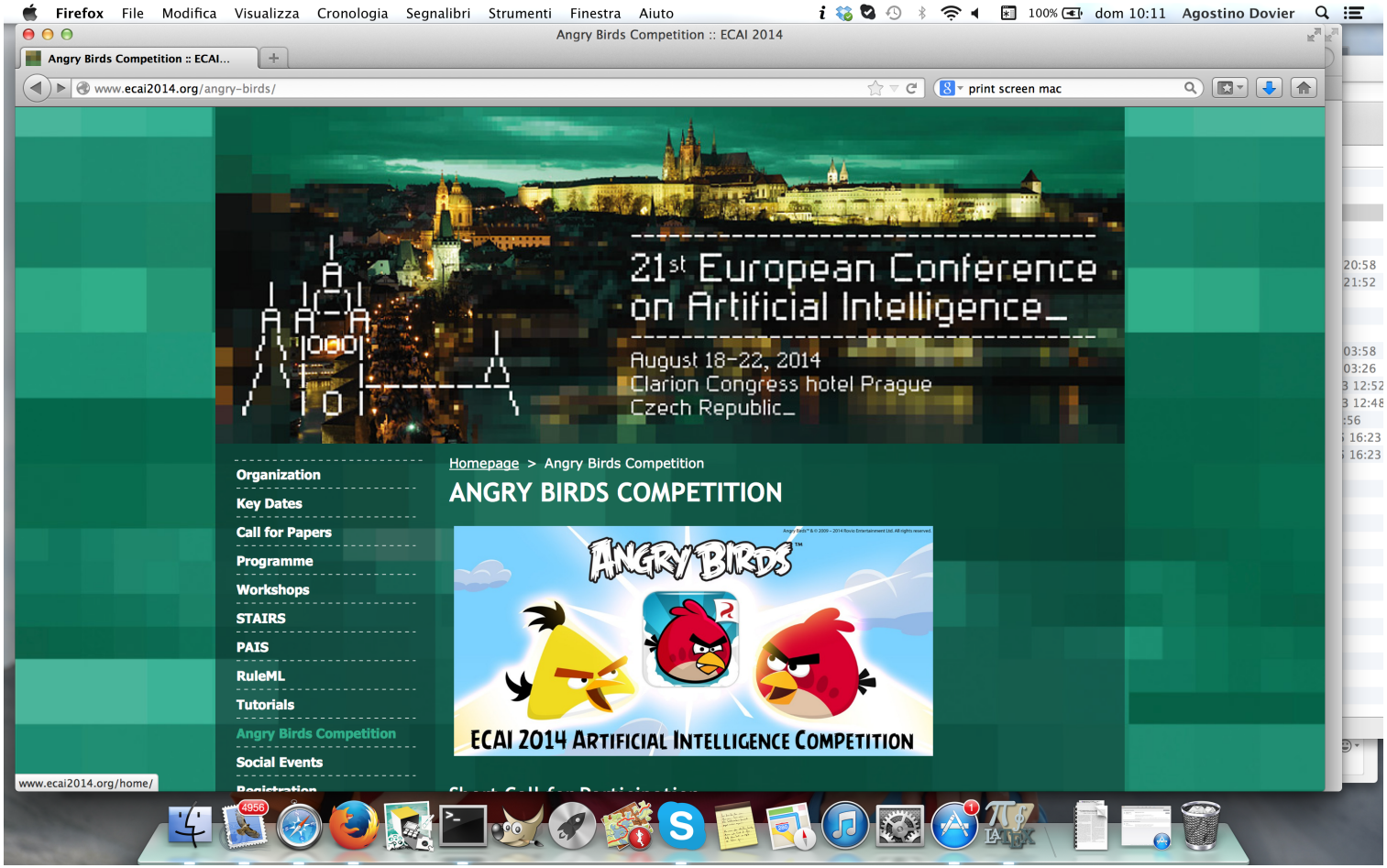
But the third remained silent.

"Can *you* tell me anything?" Abercrombie asked.

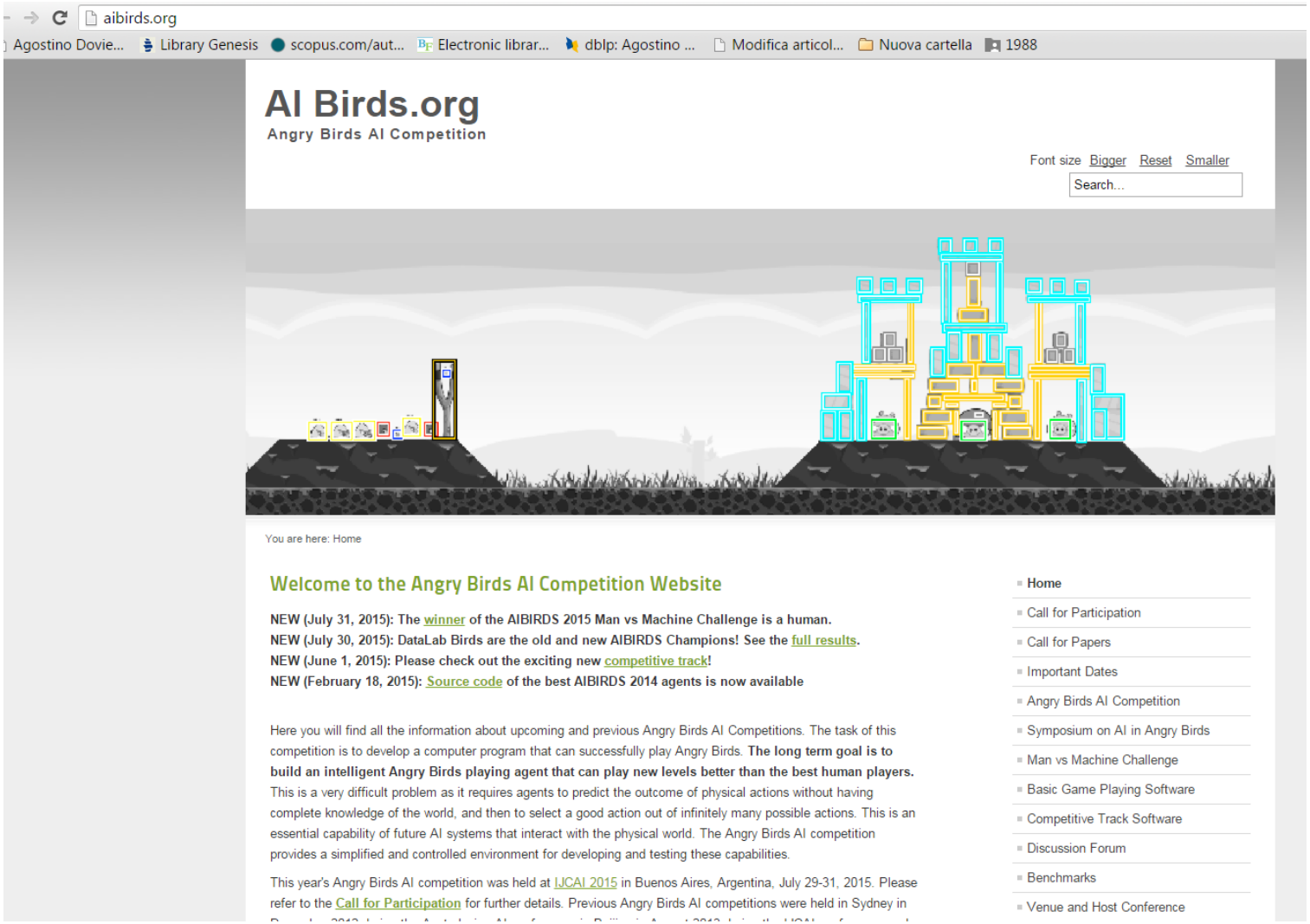
"It's funny," answered the third one with a sly smile. "At most, only one of the three of us ever tells the truth!"

Can it be deduced which of the three is the Sorcerer's Apprentice?

ANGRY BIRDS


Angry Birds Competition :: ECAI 2014
www.ecai2014.org/angry-birds/
21st European Conference on Artificial Intelligence_
August 18-22, 2014
Clarion Congress hotel Prague
Czech Republic_
Organization
Key Dates
Call for Papers
Programme
Workshops
STAIRS
PAIS
RuleML
Tutorials
Angry Birds Competition
Social Events
Homepage > Angry Birds Competition
ANGRY BIRDS COMPETITION
ANGRY BIRDS
ECAI 2014 ARTIFICIAL INTELLIGENCE COMPETITION
www.ecai2014.org/home/

Angry Birds


aibirds.org
Agostino Dovie...
Library Genesis
scopus.com/aut...
Electronic librar...
dblp: Agostino ...
Modifica articol...
Nuova cartella
1988
AI Birds.org
Angry Birds AI Competition
Font size Bigger Reset Smaller
Search...
You are here: Home
Welcome to the Angry Birds AI Competition Website
NEW (July 31, 2015): The winner of the AIBIRDS 2015 Man vs Machine Challenge is a human.
NEW (July 30, 2015): DataLab Birds are the old and new AIBIRDS Champions! See the full results.
NEW (June 1, 2015): Please check out the exciting new competitive track!
NEW (February 18, 2015): Source code of the best AIBIRDS 2014 agents is now available
Here you will find all the information about upcoming and previous Angry Birds AI Competitions. The task of this competition is to develop a computer program that can successfully play Angry Birds. The long term goal is to build an intelligent Angry Birds playing agent that can play new levels better than the best human players. This is a very difficult problem as it requires agents to predict the outcome of physical actions without having complete knowledge of the world, and then to select a good action out of infinitely many possible actions. This is an essential capability of future AI systems that interact with the physical world. The Angry Birds AI competition provides a simplified and controlled environment for developing and testing these capabilities.
This year's Angry Birds AI competition was held at IJCAI 2015 in Buenos Aires, Argentina, July 29-31, 2015. Please refer to the Call for Participation for further details. Previous Angry Birds AI competitions were held in Sydney in
Home
Call for Participation
Call for Papers
Important Dates
Angry Birds AI Competition
Symposium on AI in Angry Birds
Man vs Machine Challenge
Basic Game Playing Software
Competitive Track Software
Discussion Forum
Benchmarks
Venue and Host Conference

ASP solving (in short)

"ABSTRACT" ASP SOLVING

Let us see an abstract description of the ASP solver proposed by Yuliya Lierler (Univ. Nebraska at Omaha)

Guess who is who in the picture ...

SMODELS: ABSTRACT VIEW

For a set σ of atoms, a *record* relative to σ is an ordered set M of literals over σ, some possibly annotated by Δ, which marks them as *decision* literals. A *state* relative to σ is a record relative to σ possibly preceding symbol $\bot$. For instance, some states relative to a singleton set $\{a\}$ of atoms are

$$\emptyset, \quad a, \quad \neg a, \quad a^{\Delta}, \quad a\,\neg a, \quad \bot, \quad a\bot, \quad \neg a\bot, \quad a^{\Delta}\bot, \quad a\,\neg a\bot.$$

We say that a state is inconsistent if either $\bot$ or two complementary literals occur in it. For example, states $a\,\neg a$ and $a\bot$ are inconsistent.

By $Bodies(\Pi, a)$ we denote the set of the bodies of all rules of a regular program Π with the head a. We recall that a set U of atoms occurring in a regular program Π is *unfounded* [18, 19] on a consistent set M of literals with respect to Π if for every $a \in U$ and every $B \in Bodies(\Pi, a)$, $M \models \overline{B}$ (where B is identified with the conjunction of its elements), or $U \cap B^{pos} \neq \emptyset$.

SMODELS: ABSTRACT VIEW

For a set σ of atoms, a *record* relative to σ is an ordered set M of literals over σ, some possibly annotated by Δ, which marks them as *decision* literals. A *state* relative to σ is a record relative to σ possibly preceding symbol $\bot$. For instance, some states relative to a singleton set $\{a\}$ of atoms are

$$\emptyset, \quad a, \quad \neg a, \quad a^{\Delta}, \quad a\,\neg a, \quad \bot, \quad a\bot, \quad \neg a\bot, \quad a^{\Delta}\bot, \quad a\,\neg a\bot.$$

We say that a state is inconsistent if either $\bot$ or two complementary literals occur in it. For example, states $a\,\neg a$ and $a\bot$ are inconsistent.

By $Bodies(\Pi, a)$ we denote the set of the bodies of all rules of a regular program Π with the head a. We recall that a set U of atoms occurring in a regular program Π is *unfounded* [18, 19] on a consistent set M of literals with respect to Π if for every $a \in U$ and every $B \in Bodies(\Pi, a)$, $M \models \overline{B}$ (where B is identified with the conjunction of its elements), or $U \cap B^{pos} \neq \emptyset$.

SMODELS: ABSTRACT VIEW

Unit Propagate:

$M \implies M\ l$ if $C \vee l \in \Pi^{cl}$ and $\overline{C} \subseteq M$

Decide:

$M \implies M\ l^{\Delta}$ if l is unassigned by M

Fail:

$$M \implies \bot \text{ if } \begin{cases} M \text{ is inconsistent and different from } \bot, \\ M \text{ contains no decision literals} \end{cases}$$

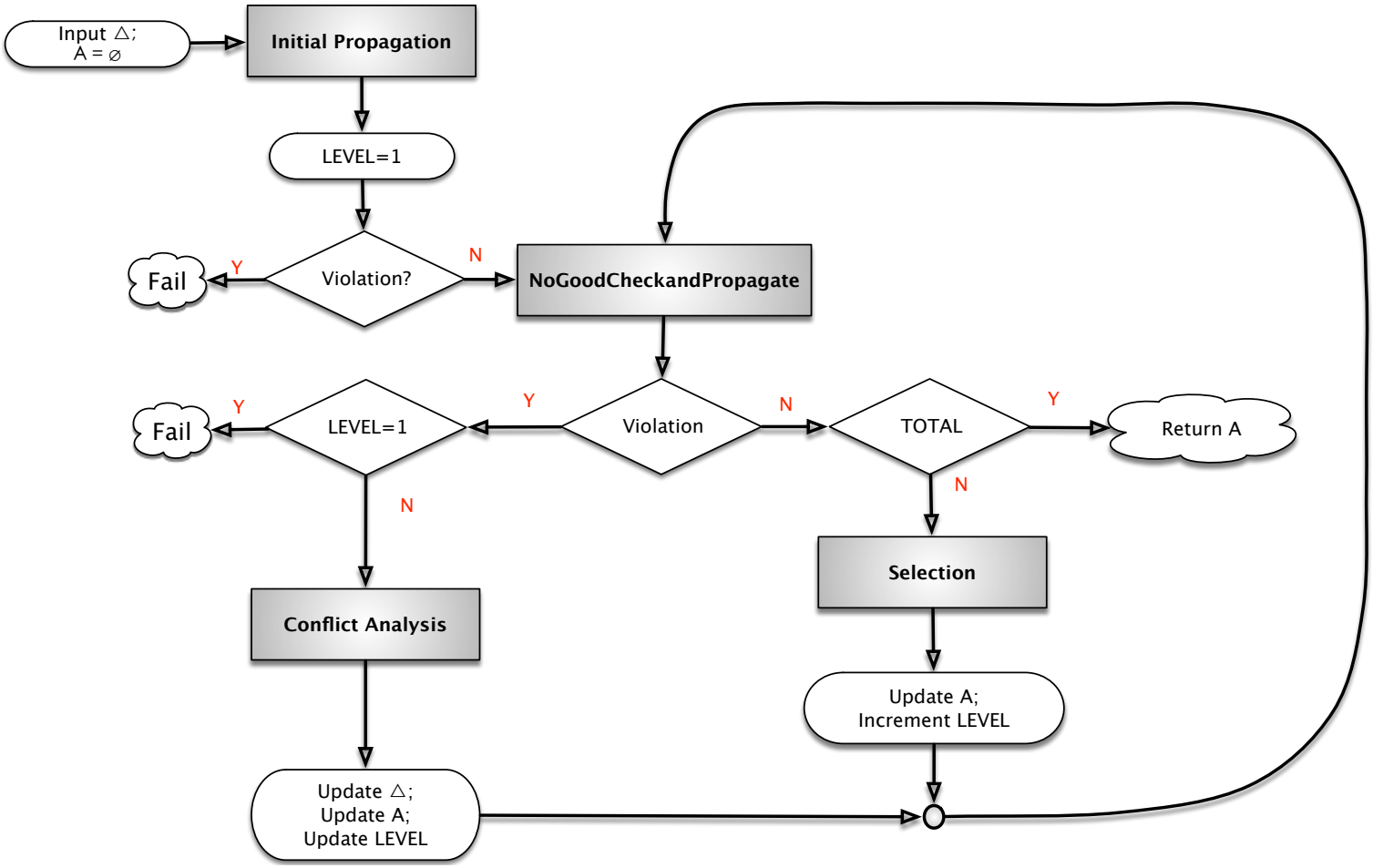
Backtrack:

$$P\ l^{\Delta}\ Q \implies P\ \bar{l} \text{ if } \begin{cases} P\ l^{\Delta}\ Q \text{ is inconsistent, and} \\ Q \text{ contains no decision literals} \end{cases}$$

Unfounded:

$M \implies M\ \neg a$ if $a \in U$ for a set U unfounded on M w.r.t. Π

CLASP (AND CUD@ASP)



The key of CLASP efficiency is conflict analysis and nogood learning