Multi-agent planning in CLP

Agostino Dovier[1] Andrea Formisano [2]
Enrico Pontelli[3]

1. Dip. Matematica e Informatica, Univ. di Udine
2. Dip. di Matematica e Informatica, Univ. di Perugia
3. Dept. Computer Science, New Mexico State Univ.

Ferrara, June 24th 2009

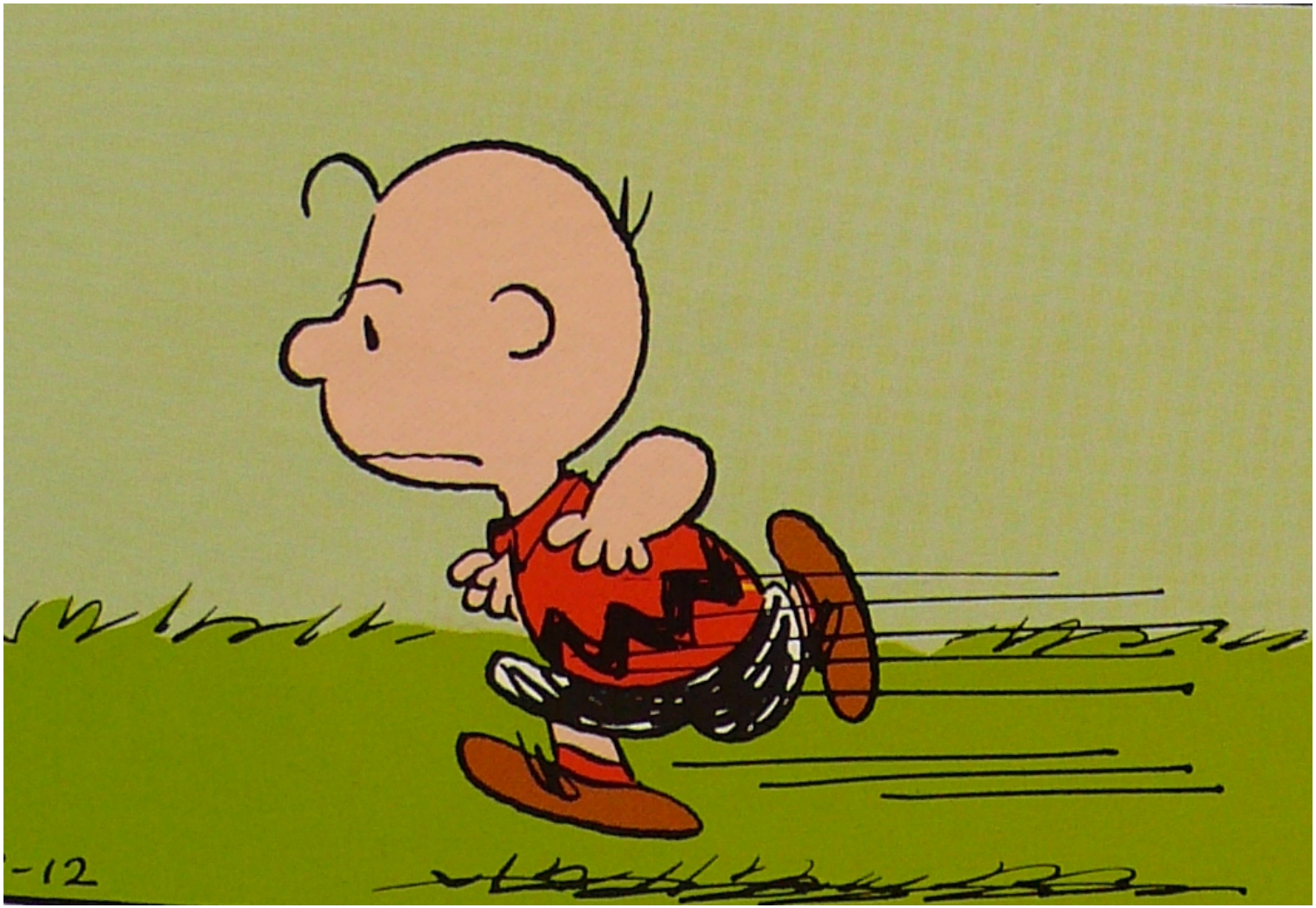
Introduction

Charlie Brown and Lucy

Charlie Brown is at the park and wishes to play football with Lucy.

Introduction

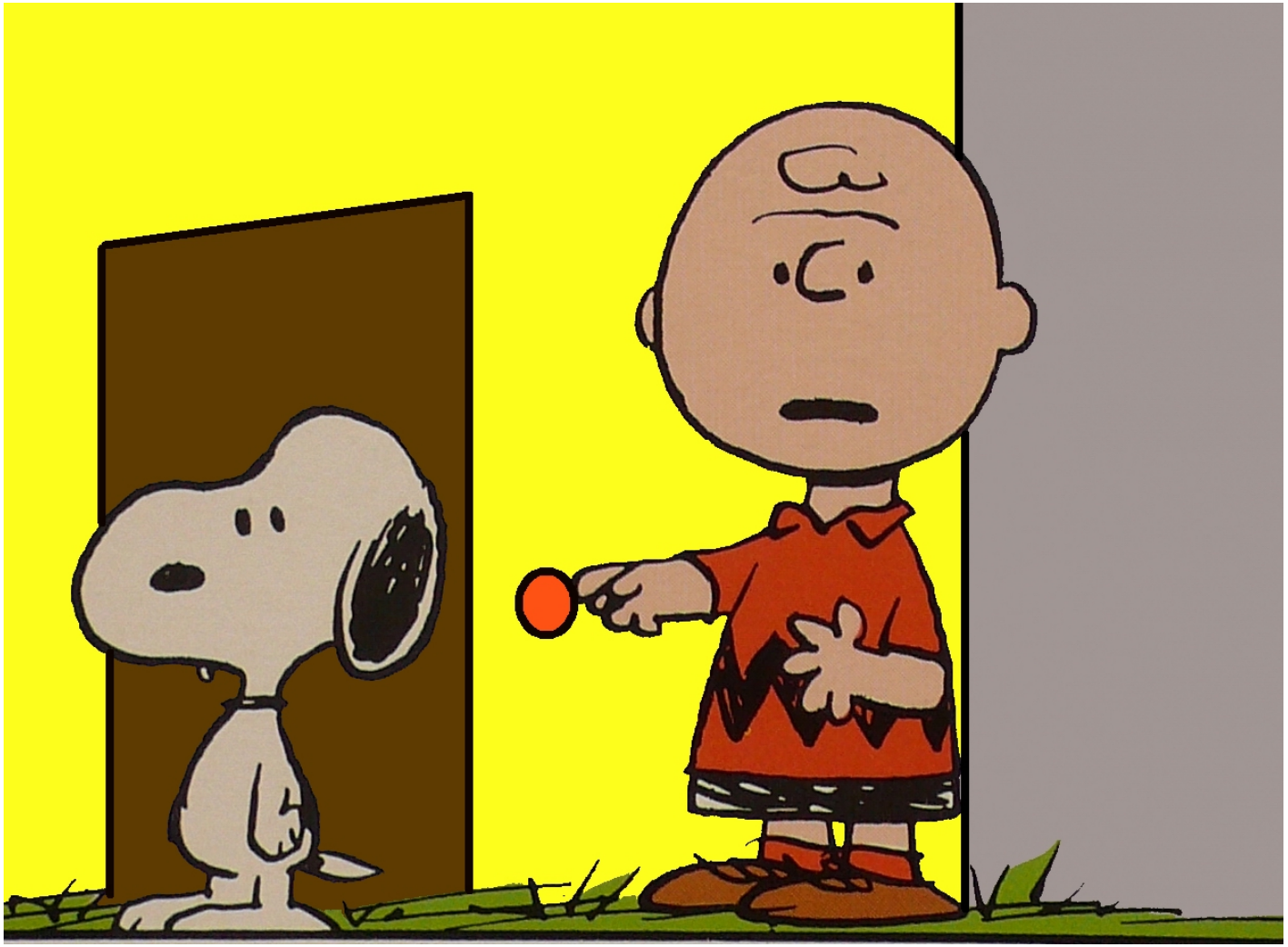
Charlie Brown and Lucy



He is running towards Lucy's home

Introduction

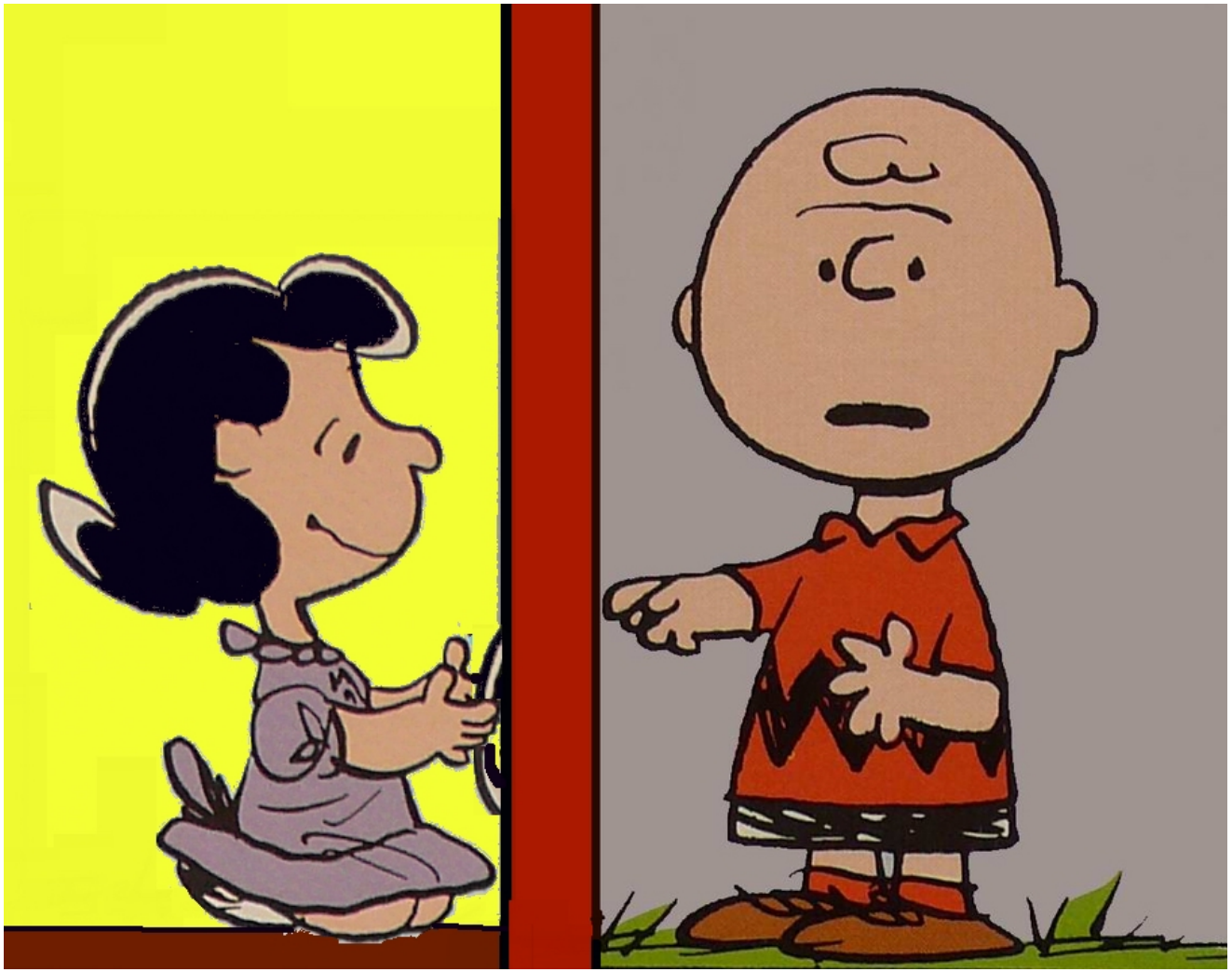
Charlie Brown and Lucy



He is ringing the bell
(Snoopy is there, but it doesn't matter)

Introduction

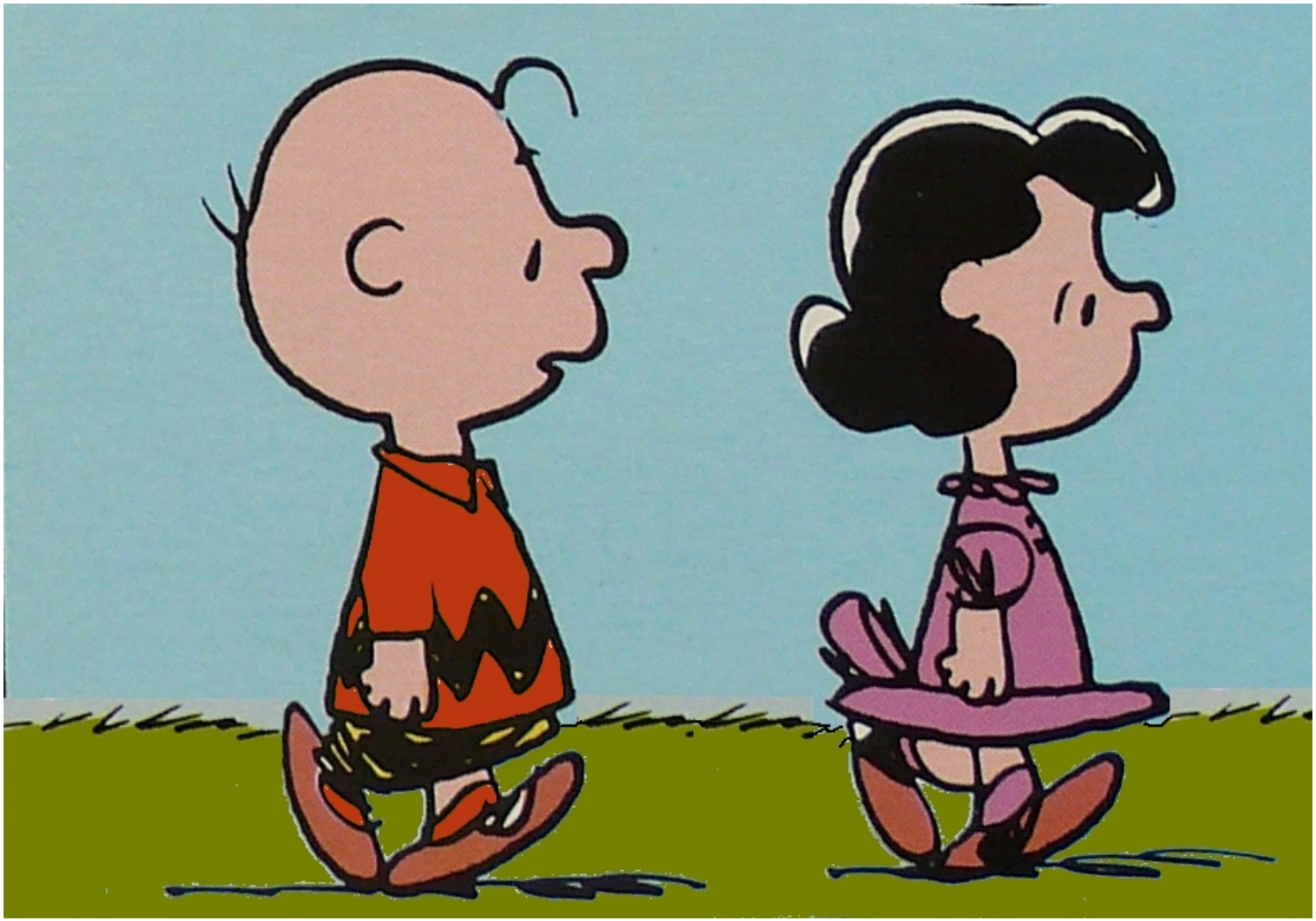
Charlie Brown and Lucy



The door is heavy:
Charlie need pushing, Lucy pulling it to open.

Introduction

Charlie Brown and Lucy



They finally can go together in the park.

Introduction

Charlie Brown and Lucy

The end is as usual, of course :)

Introduction

Charlie Brown and Lucy

Questions:

1. How many international copyright laws have I violated in the last 3 minutes?
2. How can we model this problem and inferring the plan invented by Charlie Brown?

Introduction

Charlie Brown and Lucy

Questions:

1. How many international copyright laws have I violated in the last 3 minutes?
2. How can we model this problem and inferring the plan invented by Charlie Brown?

Introduction

Charlie Brown and Lucy

Questions:

1. How many international copyright laws have I violated in the last 3 minutes?
2. How can we model this problem and inferring the plan invented by Charlie Brown?

Introduction

Encoding Charlie Brown and Lucy

- Action description languages (e.g., STRIPS, *A*, *B*, *C*, PDDL, etc) allow to model these kinds of problems
- The basic elements are
  - States, described by *Fluents* (i.e. variables –often propositional– whose values define the state)
  - Actions that allow the transition from a state to another
  - (In some languages) Static causal laws, namely constraints on a state
  - The description of the initial state and of the final state
- Often these languages are mapped on ASP.
- We mapped *B* in CLP(FD) [ICLP07]
- For this problem we need a multi-agent language with concurrent actions.

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

Let us use a Prolog-like syntax (as in *B*).

If you, instead, prefer a LISP-like syntax (as in PDDL), you are probably in the wrong conference.

Agents declaration: two agents

```
agent(charlie).
agent(lucy).
```

Three places

```
place(0).   %%% park
place(1).   %%% road
place(2).   %%% Lucy's home
```

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

Let us use a Prolog-like syntax (as in *B*).
If you, instead, prefer a LISP-like syntax (as in PDDL), you are probably in the wrong conference.

Agents declaration: two agents

```
agent(charlie).
agent(lucy).
```

Three places

```
place(0).   %%% park
place(1).   %%% road
place(2).   %%% Lucy's home
```

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

Let us use a Prolog-like syntax (as in *B*).
If you, instead, prefer a LISP-like syntax (as in PDDL), you are probably in the wrong conference.

Agents declaration: two agents

```
agent(charlie).
agent(lucy).
```

Three places

```
place(0).   %%% park
place(1).   %%% road
place(2).   %%% Lucy's home
```

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

A state is characterized by fluent values.
Fluents and their domains can be defined as follows:

```
fluent(stay(X),0,2) :- agent(X).
fluent(door,0,1).
fluent(bell,0,1).
```

fluents can also be declared *local* to agents.

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

Admissible actions:

```
action([X],move(A,B)) :-
    agent(X),
    place(A), place(B),
    1 is abs(A-B).

action([X],ring) :- agent(X).
action([X],push) :- agent(X).
action([X],pull) :- agent(X).
```

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

Executability of actions:

```
executable([X],move(0,1),[stay(X) eq 0]) :-
     agent(X).
executable([X],move(1,0),[stay(X) eq 1]) :-
     agent(X).
executable([X],move(1,2),[stay(X) eq 1, door eq 1]) :-
     agent(X).
executable([X],move(2,1),[stay(X) eq 2, door eq 1]) :-
     agent(X).
executable([X],ring,[stay(X) eq 1]) :-
     agent(X).
executable([X],push,[stay(X) eq 1, bell eq 1])  :-
     action([X],push).
executable([X],pull,[stay(X) eq 2, bell eq 1])  :-
     action([X],pull).
```

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

Action effects:

```
causes(stay(X) eq B, [actocc([X],move(A,B))]) :-
      action([X],move(A,B)).
causes(bell eq 1, [actocc([X],ring),bell eq 0]) :-
     action([X],ring).
causes(bell eq 0, [actocc([X],ring),bell eq 1]) :-
     action([X],ring).
causes(door eq 1,[actocc([X],push),actocc([Y],pull)]) :-
     action([X],push), action([Y],pull).
```

Syntax of B^{MAP}

Encoding Charlie Brown and Lucy

Initial state:

```
initially(stay(charlie) eq 0).
initially(stay(lucy) eq 2).
initially(bell eq 0).
initially(door eq 0).
```

Final state:

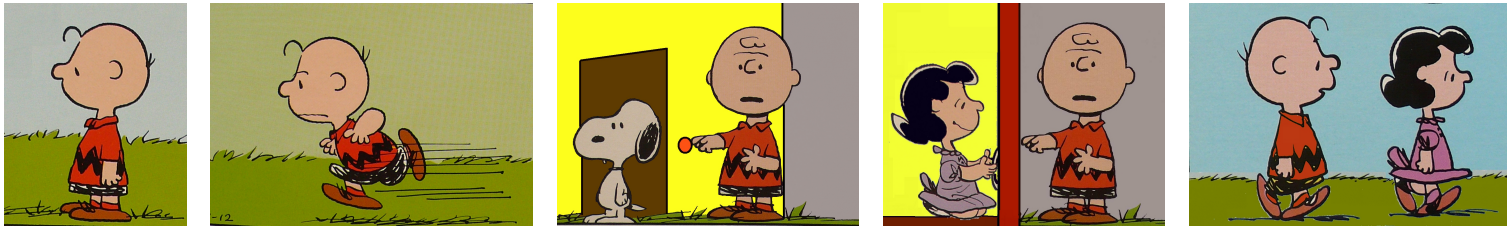
```
goal(stay(charlie) eq 0).
goal(stay(lucy) eq 0).
```

Example

Encoding Charlie Brown and Lucy

A plan (computed by B^{MAP}):

```
Time 0: stay(charlie) = 0 stay(lucy) = 2  door = 0 bell = 0
 ---->> [charlie]:move(0,1)
Time 1: stay(charlie) = 1 stay(lucy) = 2  door = 0 bell = 0
 ---->> [charlie]:ring
Time 2: stay(charlie) = 1 stay(lucy) = 2  door = 0 bell = 1
 ---->> [charlie]:push [lucy]:pull
Time 3: stay(charlie) = 1 stay(lucy) = 2  door = 1 bell = 1
 ---->> [lucy]:move(2,1)
Time 4: stay(charlie) = 1 stay(lucy) = 1  door = 1 bell = 1
 ---->> [charlie]:move(1,0) [lucy]:move(1,0)
Time 5: stay(charlie) = 0 stay(lucy) = 0  door = 1 bell = 1
```



Syntax of B^{MAP}

Some details

Let $\mathcal{F}$ be the set of fluents, $\mathcal{V}$ be the set of values, $\mathcal{A}$ be set of actions, and $\mathcal{G}$ be set of agents.
Fluents can be used in Fluent Expressions ($\mathtt{FE}$):

$$\mathtt{FE} ::= n \mid f^t \mid f \ @ \ r \mid \mathtt{FE}_1 \oplus \mathtt{FE}_2 \mid -(\mathtt{FE}) \mid \mathtt{abs}(\mathtt{FE}) \mid \mathtt{rei}(C)$$

$n \in \mathcal{V}$, $t \in \mathbb{Z}$, $\oplus \in \{+, -, *, /, \mathtt{mod}\}$, $f \in \mathcal{F}$, and $r \in \mathbb{N}$.

And in fluent constraints —propositional combinations of

$$\mathtt{FE}_1 \ \textbf{op} \ \mathtt{FE}_2$$

$\mathtt{FE}_1$, $\mathtt{FE}_2$ fluent expressions, $\textbf{op} \in \{=, \neq, \geq, \leq, >, <\}$.

f^t is a relative access to a fluent at time current$+t$
$f@r$ an absolute access to a fluent at time r.

Syntax of B^{MAP}

Some details

$\texttt{action}(Ag, x)$ where $Ag \subseteq \mathcal{G}$ and $x \in \mathcal{A}$, declares that x is meant to be executed collectively by the agents Ag.

- If $|Ag| = 1$, it is called an *individual action*.
- If $|Ag| > 1$, it is called a *collective action*.
- If $|Ag| = 0$, it represents an *exogenous action*.

The *action flag* $\texttt{actocc}(Ag, x)$, acts as a Boolean expression.
Action-fluent expressions ($\texttt{AFE}$) are defined:

$$\begin{array}{lcl} \texttt{AFE} & ::= & n \mid f^t \mid f \texttt{ @ } r \mid \texttt{actocc}(Ag, x)^t \mid \texttt{actocc}(Ag, x) \texttt{@} r \mid \\ & & \texttt{AFE}_1 \oplus \texttt{AFE}_2 \mid -(\texttt{AFE}) \mid \texttt{abs}(\texttt{AFE}) \mid \texttt{rei}(C) \end{array}$$

where $n \in \mathcal{V}$, $t \in \mathbb{Z}$, $r \in \mathbb{N}$, $f \in \mathcal{F}$, $x \in \mathcal{A}$, $Ag \subseteq \mathcal{G}$, and $\oplus \in \{+, -, *, /, \texttt{mod}\}$.

Syntax of B^{MAP}

Some details

Static causal laws are admitted: $\texttt{caused}(C_1, C_2)$ states that $C_1 \rightarrow C_2$ must be entailed by any state.

Syntactic sugar is allowed used to represent certain static causal laws frequently encountered.

$$\texttt{concurrency\_control}(C)$$

states that the action-fluent constraint C must hold (equivalent to $\texttt{caused}(\texttt{true}, C)$)

Example: two agents $\texttt{a}$ and $\texttt{b}$ can walk through a revolving door only one at the time:

$$\texttt{concurrency\_control}(\texttt{actocc}(\{a\}, \texttt{walk\_through}) + \texttt{actocc}(\{b\}, \texttt{walk\_through}) \leq 1)$$

Syntax of B^{MAP}

Some extra features

In B^{MAP} it is possible to specify information about the *cost* of each action and about the global cost of a plan.

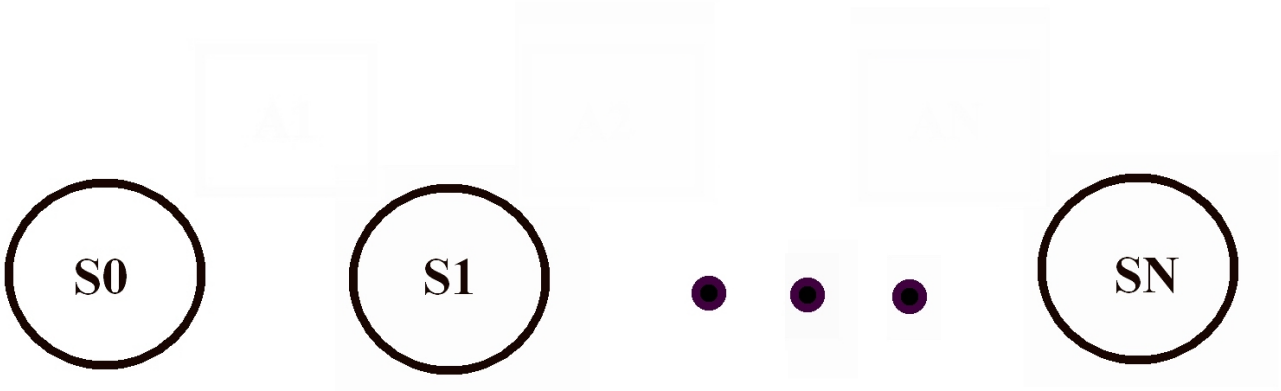
- `action_cost`(Ag, x, Val) where $Ag \subseteq \mathcal{G}$, $x \in \mathcal{A}$ (otherwise, a default cost of 1 is assigned).
- `state_cost`(FE) specifies the cost of a generic state as the result of the evaluation of the fluent expression FE, built using the fluents present in the state (otherwise, a default cost of 1 is assumed).

Semantics of B^{MAP}

- No time today for the complete semantics. You can find it in the paper (or ask us when you like).
- It is a generalization of classical Gelfond-Lifschitz semantics of the *B* language, with attention to the (expected) meaning of constraints.
- Let us focus here on its ‘constraint-based’ semantics that leads to the CLP(FD) implementation.

Implementation of B^{MAP}

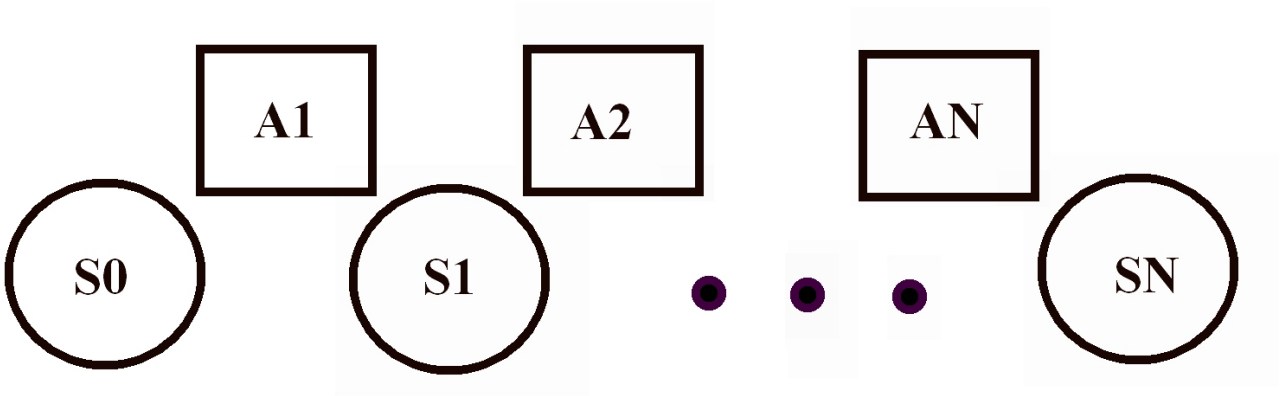
We focus on the N-plan problem.



Instances of fluent variables for each state are generated

Implementation of B^{MAP}

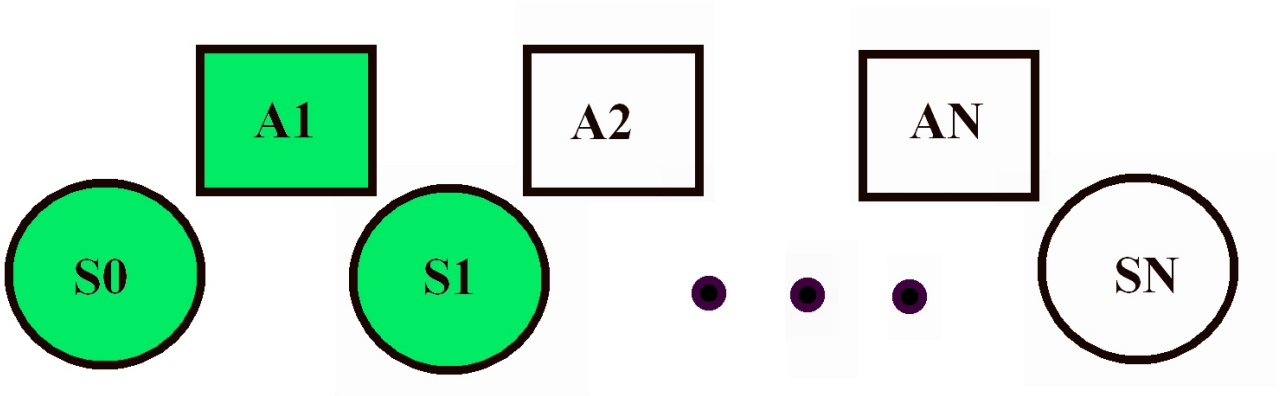
We focus on the N-plan problem.



Instances of action flags for each transition are generated

Implementation of B^{MAP}

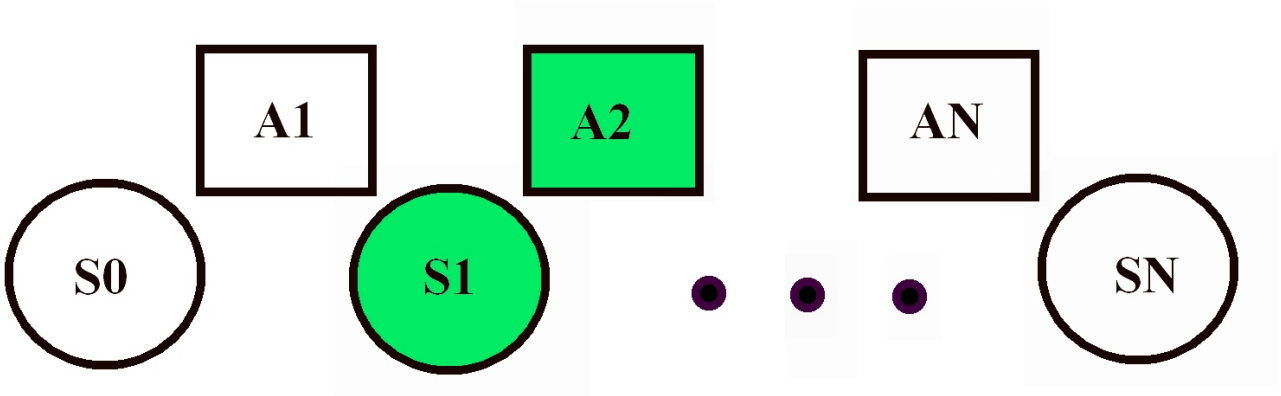
We focus on the N-plan problem.



Constraints between consecutive states and corresponding action flags are added

Implementation of B^{MAP}

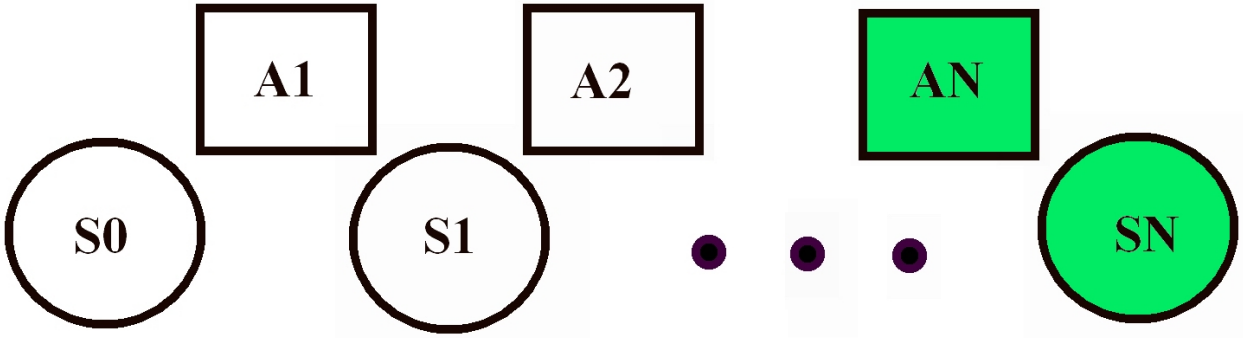
We focus on the N-plan problem.



Constraints between consecutive states and corresponding action flags are added

Implementation of B^{MAP}

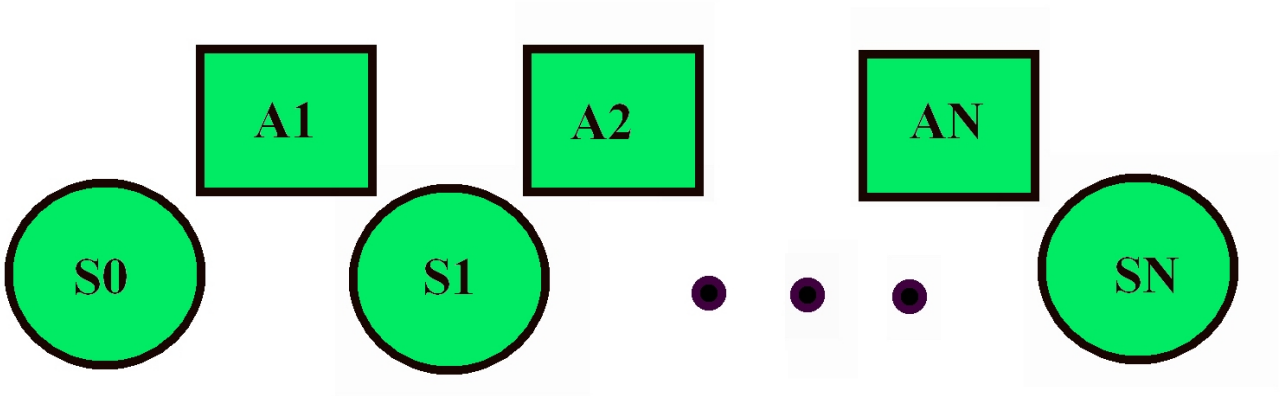
We focus on the N-plan problem.



Constraints between consecutive states and corresponding action flags are added

Implementation of B^{MAP}

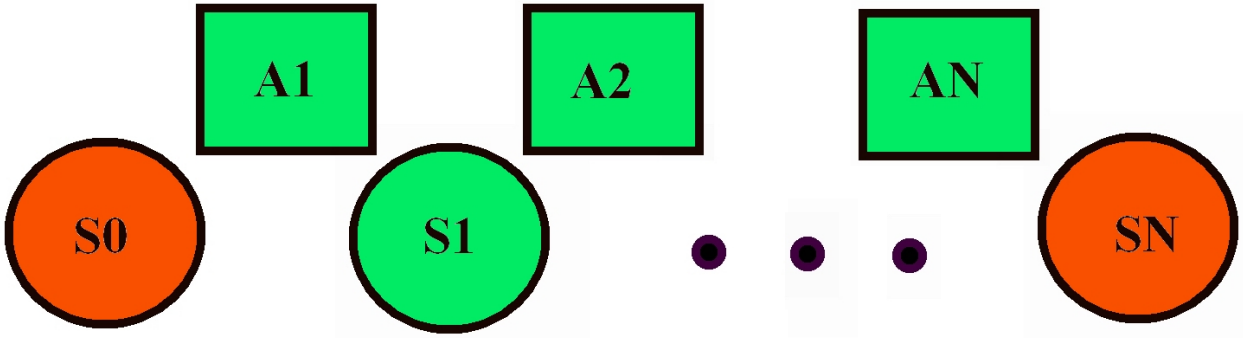
We focus on the N-plan problem.



More “global” constraints are also imposed

Implementation of B^{MAP}

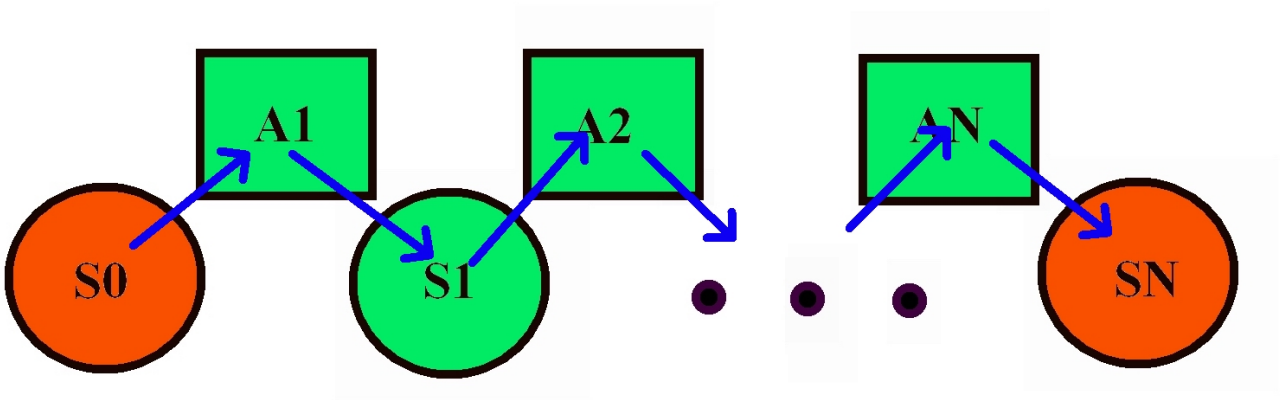
We focus on the N-plan problem.



Initial and final state information are added

Implementation of B^{MAP}

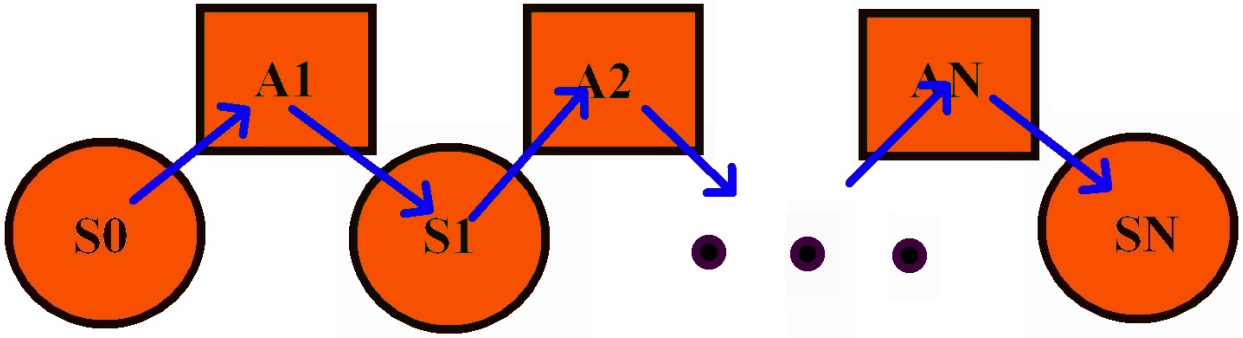
We focus on the N-plan problem.



Finally, a consistent assignment is searched using the classical propagation-labeling tree of CP.

Implementation of B^{MAP}

We focus on the N-plan problem.



If a solution is found, it is a plan

Implementation of B^{MAP}

Some constraints

Consider the state transition between states s_i and s_{i+1}. We set the domains:

- for each $f \in \mathcal{F}$, $F_f^i, F_f^{i+1} \in \mathrm{dom}(f)$;
- for each defined action Ag, x, $A_{Ag,x}^{i+1} \in \{0, 1\}$; and
- for each defined action Ag, x, and $a \in \mathcal{G}$, $G_{Ag,x,a}^{i+1} \in \{0, 1\}$.

Executability conditions are rendered by the constraint:

$$A_{Ag,x}^{i+1} \rightarrow \bigvee_{j=1}^{p_x} (C_j)*^i$$

where:
$\texttt{executable}(Ag, x, C_1), \cdots, \texttt{executable}(Ag, x, C_{p_x})$

Implementation of B^{MAP}

Some constraints

The effects of action occurrences are imposed as:

$$\bigwedge_{j=1}^{m_f} \left((PC_j)*^i \rightarrow (EC_j)^{i+1} \right)$$

for each fluent $f \in \mathcal{F}$, where $\texttt{causes}(EC_1, PC_1), \cdots, \texttt{causes}(EC_{m_f}, PC_{m_f})$ are the dynamic laws involving f.

Implementation of B^{MAP}

Some constraints

Correctness of the action-sequence is rendered by:

- $G^{i+1}_{Ag,x,a} = 0$ for each defined action Ag, x and for each agent $a \in \mathcal{G} \setminus Ag$
- $\bigwedge_{a \in Ag} G^{i+1}_{Ag,x,a} = A^{i+1}_{Ag,x}$, for each defined action Ag, x and $a \in Ag$

Each agent can execute at most one action per time can be stated as:

$$\sum_{(Ag,x)} G^{i+1}_{Ag,x,a} \leq 1$$

(for each $a \in \mathcal{G}$).

Initial and goal requirements are forced by simply imposing the constraints from `initially` and `goal` axioms.

Experiments

(codes in www.dimi.uniud.it/dovier/CLPASP/MAP)

	N	Vars	leftmost	ffc	ffcd
Three Barrels	11	62	0.12	0.11	0.07
Goat-Wolf etc.	23	219	0.14	0.04	0.28
Gas diffusion	6	132	34.9	34.9	9.65
Puzzle	15	915	62.0	64.7	4.55
Peg Solitaire	31	1999	–	–	44.7
Charlie and Lucy	5	25	0.01	0.01	0.01
Social Game	2	40	0.04	0.04	0.06
Dining philosophers	9	210	339	439	–
Fuel and Cars	10	90	736	743	0.48
Robots and Table	7	184	316	461	118

The sequential examples run significantly faster than using an ASP encoding of B.
On Peg solitaire we have surprising performances.
Not yet compared with multiagent similar languages (if any)

Conclusions

- The approach is promising and easy to be extended/modified
- We can still enhance the beauty of the syntax :)
- We will consider other constraint-based modelings
- We will study some global constraints
- We will do much more comparisons wrt other approaches

Questions?