Perspectives on Logic-based Approaches for Reasoning About Actions and Change

Agostino Dovier, Udine (IT)
Andrea Formisano, Perugia (IT)
Enrico Pontelli, NMSU (USA)

MG65 — Lexington, KY, October 26th, 2010

Motivations

- Explore alternative approaches to encode action languages (e.g., $\mathcal{B}$) in different logic programming paradigms
  - ASP
  - CLP(FD)
- Comparative experimental performance testing
- Learn lessons from CLP(FD) encoding and raise constraints to the action language—$\mathcal{B}^{MV}$

Outline

Action description languages

Research on planning requires the resolution of two key problems [Lifschitz]:

- Declarative and Elaboration Tolerant Languages to describe planning domains
- Efficient and Scalable Reasoning Algorithms

Action Description Languages are formal models to represent knowledge on actions and change (e.g., $\mathcal{A}$ and $\mathcal{B}$ [Gelfond and Lifschitz])

Specifications are given through declarative assertions that permit

- to describe actions and their effects on states
- to express queries on the underlying transition system

Action description languages

Research on planning requires the resolution of two key problems [Lifschitz]:

- Declarative and Elaboration Tolerant Languages to describe planning domains
- Efficient and Scalable Reasoning Algorithms

Action Description Languages are formal models to represent knowledge on actions and change (e.g., $\mathcal{A}$ and $\mathcal{B}$ [Gelfond and Lifschitz])

Specifications are given through declarative assertions that permit

- to describe actions and their effects on states
- to express queries on the underlying transition system

Action description languages

Research on planning requires the resolution of two key problems [Lifschitz]:

- Declarative and Elaboration Tolerant Languages to describe planning domains
- Efficient and Scalable Reasoning Algorithms

Action Description Languages are formal models to represent knowledge on actions and change (e.g., $\mathcal{A}$ and $\mathcal{B}$ [Gelfond and Lifschitz])

Specifications are given through declarative assertions that permit

- to describe actions and their effects on states
- to express queries on the underlying transition system

A *planning domain* $\mathcal{D}$ can be described through an action domain description, which defines the notions of

FLUENTS i.e., *variables* describing the state of the world, and whose value can change

STATES i.e., possible configurations of the domain of interest: an assignment of values to the fluents

ACTIONS that affect the state of the world, and thus cause the transition from a state to another

A Planning Problem $P = \langle \mathcal{D}, I, O \rangle$ includes

- Description (complete or partial) of the Initial state
- Description of the Final state

The language $\mathcal{B}$

Action Description

Let `a` be an action and ℓ be a Boolean literal. We have:

- Executability conditions:
  `executable(a, [list-of-preconditions])`
  asserting that the given preconditions have to be satisfied in the current state for the action `a` to be executable
- Dynamic causal laws:
  `causes(a, ℓ, [list-of-preconditions])`
  describes the effect (the fluent literal ℓ) of the execution of action `a` in a state satisfying the given preconditions
- Static causal laws:
  `caused([list-of-preconditions], ℓ)`
  describes the fact that the fluent literal ℓ is true in a state satisfying the given preconditions

- *Initial state*

$$\texttt{initially}(\ell)$$

asserts that ℓ holds in the initial state.

- *Goal*

$$\texttt{goal}(\ell)$$

asserts that ℓ is required to hold in the final state.

ASP Encoding of $\mathcal{B}$

Ideas from [Gelfond and Lifschitz 92]

We designed a Prolog program that translate an action description $\mathcal{D}$, with initial and final constraints $\mathcal{O}$ and a plan length N, in an ASP program $\Pi_{\mathcal{D}}(\mathrm{N}, \mathcal{O})$.

time(0..N)

fluent(f). action(a).

literal(F): $-$fluent(F).
complement(F, neg(F)).

literal(neg(F)): $-$fluent(F).
complement(neg(F), F).

ASP Encoding of $\mathcal{B}$

Ideas from [Gelfond and Lifschitz 92]

We designed a Prolog program that translate an action description $\mathcal{D}$, with initial and final constraints $\mathcal{O}$ and a plan length N, in an ASP program $\Pi_{\mathcal{D}}(\mathrm{N}, \mathcal{O})$.

time(0..N)

fluent(f). action(a).

literal(F): $-$fluent(F).
complement(F, neg(F)).

literal(neg(F)): $-$fluent(F).
complement(neg(F), F).

ASP Encoding of $\mathcal{B}$

Ideas from [Gelfond and Lifschitz 92]

We designed a Prolog program that translate an action description $\mathcal{D}$, with initial and final constraints $\mathcal{O}$ and a plan length N, in an ASP program $\Pi_{\mathcal{D}}(\mathrm{N}, \mathcal{O})$.

$$\mathrm{time}(0..\mathrm{N})$$

$$\mathrm{fluent}(f). \qquad \mathrm{action}(a).$$

$$\mathrm{literal}(F) : -\mathrm{fluent}(F). \qquad \mathrm{literal}(\mathrm{neg}(F)) : -\mathrm{fluent}(F).$$
$$\mathrm{complement}(F, \mathrm{neg}(F)). \qquad \mathrm{complement}(\mathrm{neg}(F), F).$$

ASP Encoding of $\mathcal{B}$

Ideas from [Gelfond and Lifschitz 92]

We designed a Prolog program that translate an action description $\mathcal{D}$, with initial and final constraints $\mathcal{O}$ and a plan length N, in an ASP program $\Pi_{\mathcal{D}}(\mathrm{N}, \mathcal{O})$.

$$\mathrm{time}(0..\mathrm{N})$$

$$\mathrm{fluent}(f). \qquad \mathrm{action}(a).$$

$$\mathrm{literal}(F) \text{:} -\mathrm{fluent}(F). \qquad \mathrm{literal}(\mathrm{neg}(F)) \text{:} -\mathrm{fluent}(F).$$
$$\mathrm{complement}(F, \mathrm{neg}(F)). \qquad \mathrm{complement}(\mathrm{neg}(F), F).$$

ASP Encoding of $\mathcal{B}$

The predicate holds(FluentLiteral,Time) is defined using the axioms:

- $\texttt{executable}(a, [\ell^1_1, \ldots, \ell^1_{r_1}]), \ldots, \texttt{executable}(a, [\ell^h_1, \ldots, \ell^h_{r_h}])$:

$$\text{possible}(a, T) \text{: } -\text{time}(T), \text{holds}(\ell^1_1, T), \ldots, \text{holds}(\ell^1_{r_1}, T).$$
$$\ldots$$
$$\text{possible}(a, T) \text{: } -\text{time}(T), \text{holds}(\ell^h_1, T), \ldots, \text{holds}(\ell^h_{r_h}, T).$$

- Static causal laws: $\texttt{caused}([\ell_1, \ldots, \ell_r], \ell)$:

$$\text{holds}(\ell, T) \text{: } -\text{time}(T), \text{holds}(\ell_1, T), \ldots, \text{holds}(\ell_r, T).$$

- Dynamic causal laws: $\texttt{causes}(a, \ell, [\ell_1, \ldots, \ell_r])$:

$$\begin{aligned} \text{holds}(\ell, T+1) \quad :- \quad & \text{time}(T), \text{occ}(a, T), \\ & \text{holds}(\ell_1, T), \ldots, \text{holds}(\ell_r, T). \end{aligned}$$

ASP Encoding of $\mathcal{B}$

The predicate holds(FluentLiteral,Time) is defined using the axioms:

- $\texttt{executable}(a, [\ell^1_1, \ldots, \ell^1_{r_1}]), \ldots, \texttt{executable}(a, [\ell^h_1, \ldots, \ell^h_{r_h}])$:

$$\text{possible}(a, T) \text{: } -\text{time}(T), \text{holds}(\ell^1_1, T), \ldots, \text{holds}(\ell^1_{r_1}, T).$$
$$\ldots$$
$$\text{possible}(a, T) \text{: } -\text{time}(T), \text{holds}(\ell^h_1, T), \ldots, \text{holds}(\ell^h_{r_h}, T).$$

- Static causal laws: $\texttt{caused}([\ell_1, \ldots, \ell_r], \ell)$:

$$\text{holds}(\ell, T) \text{: } -\text{time}(T), \text{holds}(\ell_1, T), \ldots, \text{holds}(\ell_r, T).$$

- Dynamic causal laws: $\texttt{causes}(a, \ell, [\ell_1, \ldots, \ell_r])$:

$$\begin{aligned} \text{holds}(\ell, T+1) \quad :- \quad & \text{time}(T), \text{occ}(a, T), \\ & \text{holds}(\ell_1, T), \ldots, \text{holds}(\ell_r, T). \end{aligned}$$

ASP Encoding of $\mathcal{B}$

The predicate holds(FluentLiteral,Time) is defined using the axioms:

- $\texttt{executable}(a, [\ell^1_1, \ldots, \ell^1_{r_1}]), \ldots, \texttt{executable}(a, [\ell^h_1, \ldots, \ell^h_{r_h}])$:

$$\text{possible}(a, T) \text{:} -\text{time}(T), \text{holds}(\ell^1_1, T), \ldots, \text{holds}(\ell^1_{r_1}, T).$$
$$\ldots$$
$$\text{possible}(a, T) \text{:} -\text{time}(T), \text{holds}(\ell^h_1, T), \ldots, \text{holds}(\ell^h_{r_h}, T).$$

- Static causal laws: $\texttt{caused}([\ell_1, \ldots, \ell_r], \ell)$:

$$\text{holds}(\ell, T) \text{:} -\text{time}(T), \text{holds}(\ell_1, T), \ldots, \text{holds}(\ell_r, T).$$

- Dynamic causal laws: $\texttt{causes}(a, \ell, [\ell_1, \ldots, \ell_r])$:

$$\begin{aligned} \text{holds}(\ell, T+1) \quad :- \quad & \text{time}(T), \text{occ}(a, T), \\ & \text{holds}(\ell_1, T), \ldots, \text{holds}(\ell_r, T). \end{aligned}$$

ASP Encoding of $\mathcal{B}$

- State consistency:

$$:-\text{time}(T), \text{fluent}(F), \text{holds}(F, T), \text{holds}(\text{neg}(F), T).$$

- Frame problem:

$$\begin{aligned}\text{holds}(L, T+1) \quad :- \quad & \text{time}(T), \text{literal}(L), \text{holds}(L, T), \\ & \text{complement}(L, L_1), \textbf{not}\ \text{holds}(L_1, T+1).\end{aligned}$$

- Initial state and goal:

$$\text{holds}(L, 0) :- \text{initially}(L). \qquad :- \text{goal}(L), \textbf{not}\ \text{holds}(L, \text{N}).$$

- One action per time:

$$\begin{aligned}& 1\{\text{occ}(A, T) : \text{action}(A)\}1 :- \text{time}(T), T < \text{N}. \\ & :- \text{action}(A), \text{time}(T), \text{occ}(A, T), \textbf{not}\ \text{possible}(A, T).\end{aligned}$$

In absence of static laws a simplified mapping has been implemented (leading to smaller ASP programs, but not always to faster executions)

ASP Encoding of $\mathcal{B}$

- State consistency:

$$:-\text{time}(T), \text{fluent}(F), \text{holds}(F, T), \text{holds}(\text{neg}(F), T).$$

- Frame problem:

$$\begin{array}{lll} \text{holds}(L, T+1) & :- & \text{time}(T), \text{literal}(L), \text{holds}(L, T), \\ & & \text{complement}(L, L_1), \textbf{not}\ \text{holds}(L_1, T+1). \end{array}$$

- Initial state and goal:

$$\text{holds}(L, 0) :- \text{initially}(L). \qquad :- \text{goal}(L), \textbf{not}\ \text{holds}(L, \text{N}).$$

- One action per time:

$$1\{\text{occ}(A, T) : \text{action}(A)\}1 :- \text{time}(T), T < \text{N}.$$
$$:- \text{action}(A), \text{time}(T), \text{occ}(A, T), \textbf{not}\ \text{possible}(A, T).$$

In absence of static laws a simplified mapping has been implemented (leading to smaller ASP programs, but not always to faster executions)

ASP Encoding of $\mathcal{B}$

- State consistency:

$$:-\text{time}(T), \text{fluent}(F), \text{holds}(F, T), \text{holds}(\text{neg}(F), T).$$

- Frame problem:

$$\begin{array}{lll} \text{holds}(L, T+1) & :- & \text{time}(T), \text{literal}(L), \text{holds}(L, T), \\ & & \text{complement}(L, L_1), \textbf{not}\ \text{holds}(L_1, T+1). \end{array}$$

- Initial state and goal:

$$\text{holds}(L, 0) :- \text{initially}(L). \qquad :- \text{goal}(L), \textbf{not}\ \text{holds}(L, \text{N}).$$

- One action per time:

$$1\{\text{occ}(A, T) : \text{action}(A)\}1 :- \text{time}(T), T < \text{N}.$$
$$:- \text{action}(A), \text{time}(T), \text{occ}(A, T), \textbf{not}\ \text{possible}(A, T).$$

In absence of static laws a simplified mapping has been implemented (leading to smaller ASP programs, but not always to faster executions)

ASP Encoding of $\mathcal{B}$

- State consistency:

$$:-\text{time}(T), \text{fluent}(F), \text{holds}(F, T), \text{holds}(\text{neg}(F), T).$$

- Frame problem:

$$\begin{array}{lll} \text{holds}(L, T+1) & :- & \text{time}(T), \text{literal}(L), \text{holds}(L, T), \\ & & \text{complement}(L, L_1), \textbf{not}\ \text{holds}(L_1, T+1). \end{array}$$

- Initial state and goal:

$$\text{holds}(L, 0) :- \text{initially}(L). \qquad :- \text{goal}(L), \textbf{not}\ \text{holds}(L, \text{N}).$$

- One action per time:

$$1\{\text{occ}(A, T) : \text{action}(A)\}1 :- \text{time}(T), T < \text{N}.$$
$$:- \text{action}(A), \text{time}(T), \text{occ}(A, T), \textbf{not}\ \text{possible}(A, T).$$

In absence of static laws a simplified mapping has been implemented (leading to smaller ASP programs, but not always to faster executions)

ASP Encoding of $\mathcal{B}$

- State consistency:

$$:-\text{time}(T), \text{fluent}(F), \text{holds}(F, T), \text{holds}(\text{neg}(F), T).$$

- Frame problem:

$$\begin{array}{lll} \text{holds}(L, T+1) & :- & \text{time}(T), \text{literal}(L), \text{holds}(L, T), \\ & & \text{complement}(L, L_1), \textbf{not}\ \text{holds}(L_1, T+1). \end{array}$$

- Initial state and goal:

$$\text{holds}(L, 0) :- \text{initially}(L). \qquad :- \text{goal}(L), \textbf{not}\ \text{holds}(L, \text{N}).$$

- One action per time:

$$1\{\text{occ}(A, T) : \text{action}(A)\}1 :- \text{time}(T), T < \text{N}.$$
$$:- \text{action}(A), \text{time}(T), \text{occ}(A, T), \textbf{not}\ \text{possible}(A, T).$$

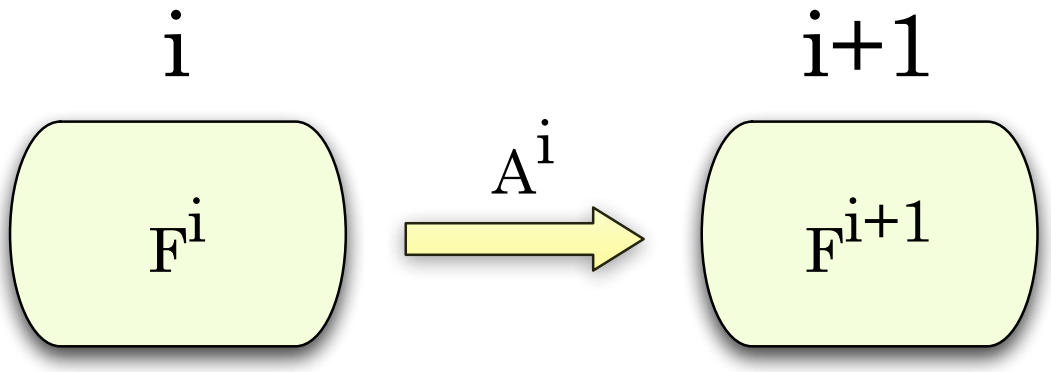
In absence of static laws a simplified mapping has been implemented (leading to smaller ASP programs, but not always to faster executions)

Modeling $\mathcal{B}$ in CLP(FD)

- Action descriptions are mapped to finite domain constraints
- Constrained variables are introduced for fluents and action occurrences
- Executability conditions and causal laws are rendered by imposing constraints
- Solutions of the constrains identify plans and trajectories

Modelling $\mathcal{B}$ in CLP(FD)

Main idea



For all states $s_0, \ldots, s_N$

- Every fluent f is represented as a Boolean variable F^i.
- Every action a is represented as a Boolean variable A^i ($i < N$).

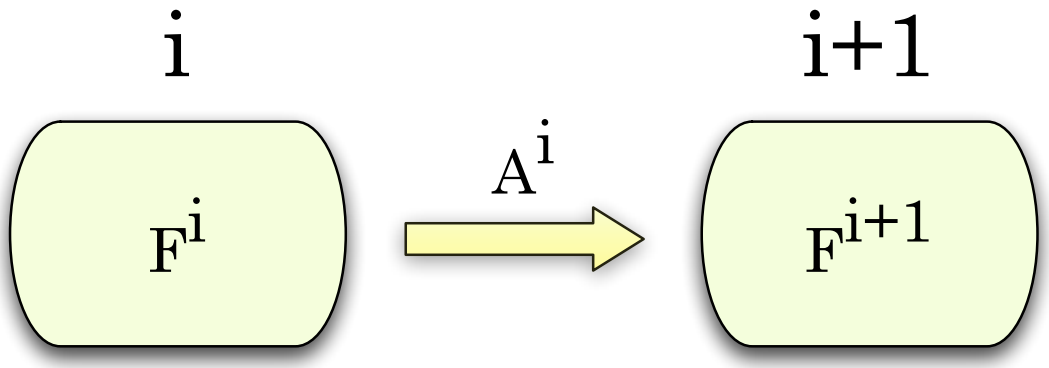
$F^{i+1} = 1 \Leftrightarrow$

One action a that set F to true is fired ($A^i = 1$), or

None of the actions a that set F to false are fired and $F^i = 1$

Modelling $\mathcal{B}$ in CLP(FD)

Main idea



For all states $s_0, \ldots, s_N$

- Every fluent f is represented as a Boolean variable F^i.
- Every action a is represented as a Boolean variable A^i ($i < N$).

$F^{i+1} = 1 \Leftrightarrow$

One action a that set F to true is fired ($A^i = 1$), or
None of the actions a that set F to false are fired and $F^i = 1$

Modelling $\mathcal{B}$ in CLP(FD)

Some constraints

$$F^{i+1} = 1 \equiv \mathsf{PosFired}^i_f = 1 \vee (\mathsf{NegFired}^i_f = 0 \wedge F^i = 1) \quad (1)$$

$$\mathsf{PosFired}^i_f = 0 \vee \mathsf{NegFired}^i_f = 0 \quad (2)$$

$$\mathsf{PosFired}^i_f = 1 \equiv \mathsf{PosDyn}^i_f = 1 \vee \mathsf{PosStat}^{i+1}_f = 1 \quad (3)$$

$$\mathsf{PosDyn}^i_f = 1 \equiv \bigvee_{j=1}^{m} (\hat{\alpha}^i_j \wedge A^{i+1}_{t_j} = 1) \quad (4)$$

$$\mathsf{PosStat}^i_f = 1 \equiv \bigvee_{j=1}^{h} \hat{\delta}^i_j \quad (5)$$

$$\mathsf{NegFired}^i_f = 1 \equiv \mathsf{NegDyn}^i_f = 1 \vee \mathsf{NegStat}^{i+1}_f = 1 \quad (6)$$

$$\mathsf{NegDyn}^i_f = 1 \equiv \bigvee_{j=1}^{p} (\hat{\beta}^i_j \wedge A^{i+1}_{z_j} = 1) \quad (7)$$

$$\mathsf{NegStat}^i_f = 1 \equiv \bigvee_{j=1}^{k} \hat{\gamma}^i_j \quad (8)$$

$$A^{i+1}_j = 1 \rightarrow \bigvee_{j=1}^{q} \hat{\eta}^i_{r_j} \quad (9)$$

$$\sum_{a_j \in \mathcal{A}} A^i_j = 1 \quad (10)$$

THE LANGUAGE $\mathcal{B}^{MV}$

FLUENTS: introduced through *domain declarations*:

$$\texttt{fluent}(\texttt{f},\texttt{v}_1,\texttt{v}_2),\quad \texttt{fluent}(\texttt{f},\texttt{Set})$$

ANNOTATED FLUENTS: modeling backward references:

$$\texttt{f}^{-\texttt{a}} \qquad \text{with } a \in \mathbb{N}$$

FLUENT EXPRESSIONS:

$$\texttt{FE} ::= \texttt{n} \mid \texttt{AF} \mid \texttt{abs}(\texttt{FE}) \mid \texttt{FE}_1 \oplus \texttt{FE}_2 \mid \texttt{rei}(\texttt{FC})$$

with $\oplus \in \{+, -, *, /, \texttt{mod}\}$

FLUENT CONSTRAINTS:

$$\texttt{FC} ::= \texttt{FE}_1 \ \texttt{op} \ \texttt{FE}_2$$

with $\texttt{op} \in \{\texttt{eq},\texttt{neq},\texttt{geq},\texttt{leq},\texttt{lt},\texttt{gt}\}$

- Dynamic causal laws

  `causes(a, C1, C)`

- Static causal laws

  `caused(C, C1)`

- Executability conditions

  `executable(a, C)`

where `a` is an action, C_1 is a fluent constraint, and C a conjunction of fluent constraints.

The language $\mathcal{B}^{MV}$

Main advantages

- It allows the compact representation of numerical domains.
- An encoding in $\mathcal{B}$ is still possible, but the number of fluents explodes
- Consider

$$\texttt{causes}(a, f = f^{-1} + 1, [])$$

with $\text{dom}(f) = [1..100]$.

  - In $\mathcal{B}$:

$$\texttt{causes}(a, f_2, [f_1]). \quad \ldots \quad \texttt{causes}(a, f_{100}, [f_{99}])$$

  - Alternative encodings (e.g., bit-based) incurs analogous problems
- The CLP(FD) mapping requires minor changes.

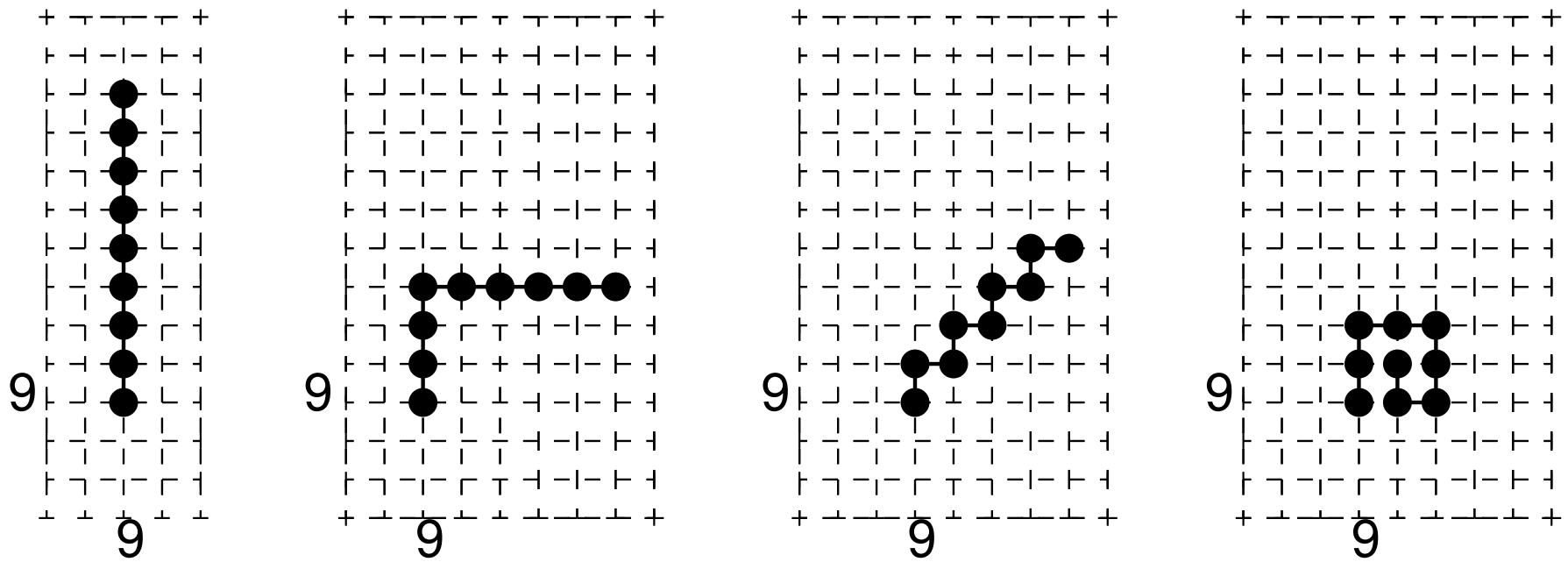
Experimental comparison

Benchmarks used

- *Hydraulic planning* (by Michael Gelfond et al., ASP 2009)
- *Peg Solitaire* (ASP 2009)
- *Sam Lloyd's 15 puzzle* (ASP 2009)
- *Towers of Hanoi* (ASP 2009)
- The *trucks* domain from the (IPC5)
- A generalized version of the classical *3-barrels* problem
- The *Gas Diffusion problem*
- The *reverse folding problem*.
- The *Tangram* puzzle.

Experimental comparison

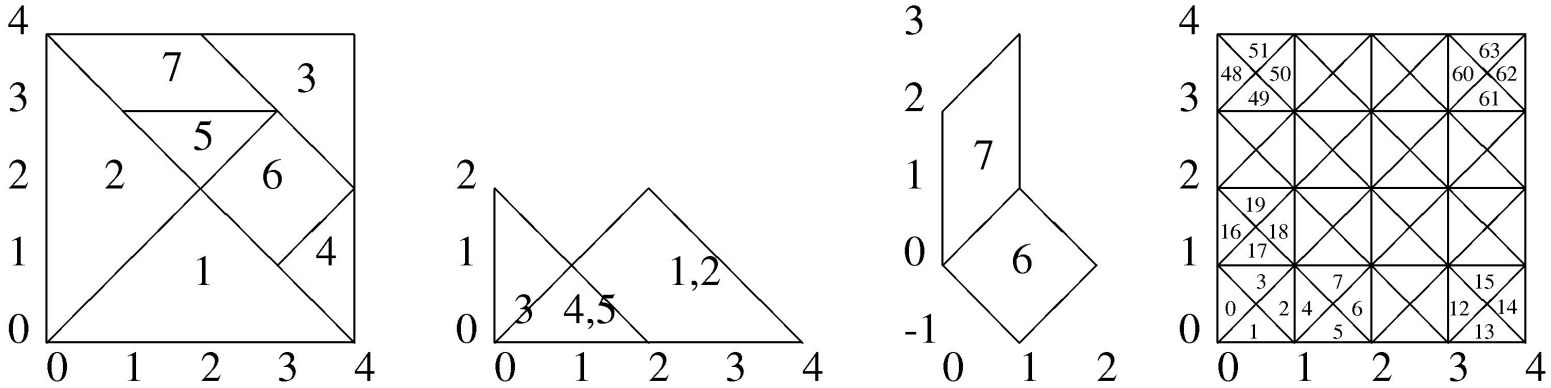
Benchmarks used: Reverse folding problem



Four foldings with k=9: The initial (straight line) folding, the result of a clockwise pivot move on the 4th element, a zigzag folding, and a spiral folding.

Experimental Comparison

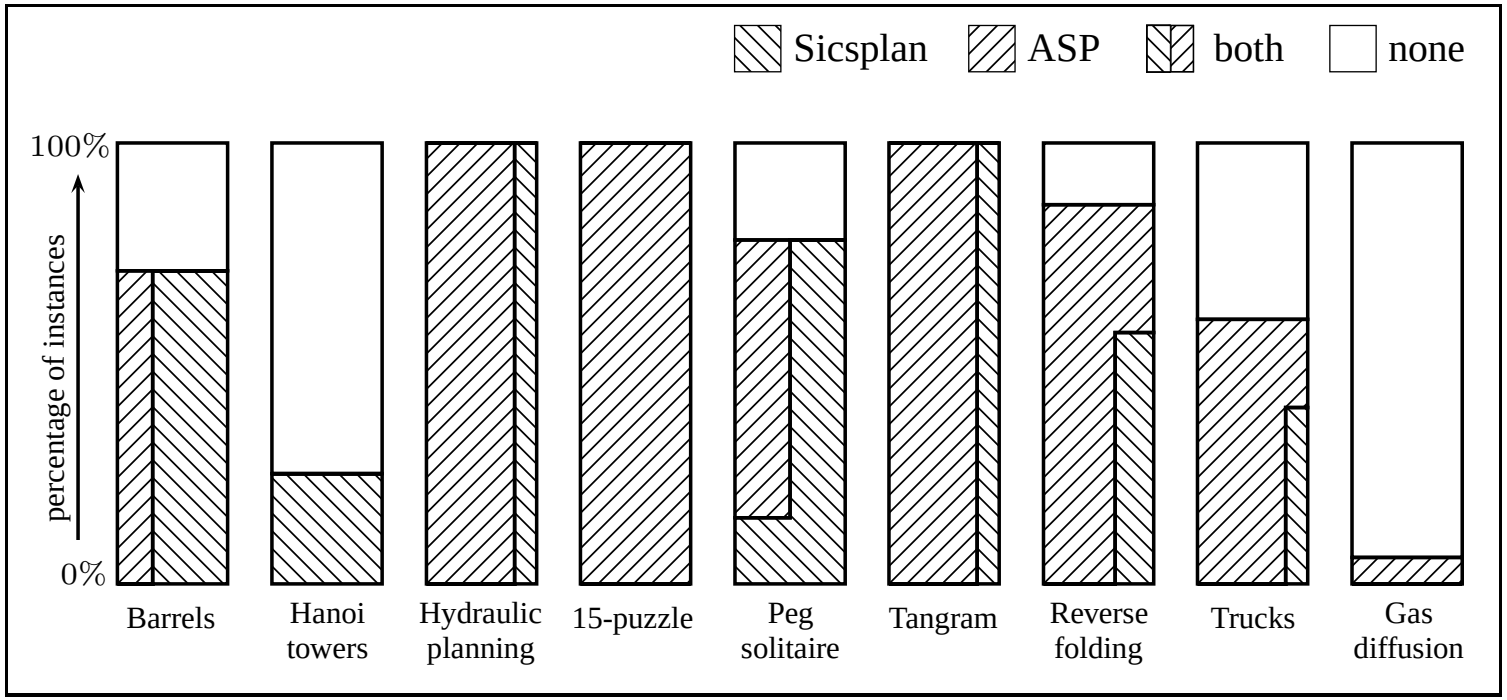
Benchmarks used: Tangram



Tangram solution, “Base” position of the seven blocks, and space discretization using triangles.

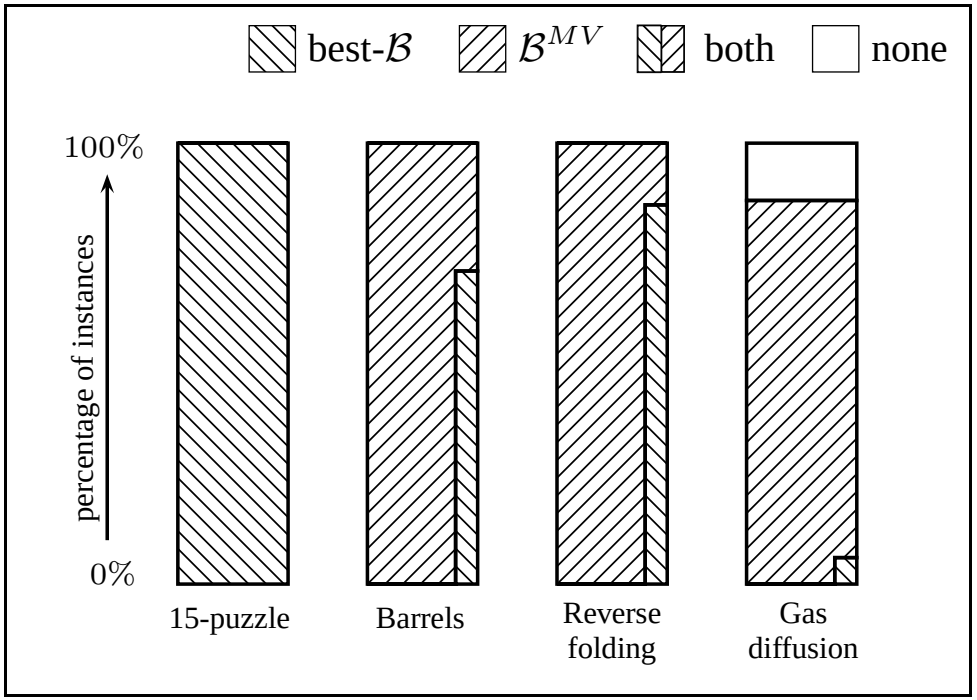
Experimental comparison

ASP (clasp) vs CLP (SICStus)



Experimental comparison

$\mathcal{B}$ best vs $\mathcal{B}^{MV}$(CLP)



Discussion

Other things already done

- Other constraints (*cost* of each action, fluent, and about the global cost of a plan, capabilities of plan minimization, temporal fluent literals, integrity constraints, . . .) [TPLP 2010]
- Multiagent reasoning (with central coordination) [FUIN 2011]
- Nicer syntax

Discussion

Things to do

- Looking into the future: reasoning/constraints referring to future steps of the trajectory
- Preferences (qualitative, quantitative) in action execution
- Distributed multiagent reasoning
- Heuristics

WEBSITE http://www.dimi.uniud.it/dovier/CLPASP

Discussion

Things to do

- Looking into the future: reasoning/constraints referring to future steps of the trajectory
- Preferences (qualitative, quantitative) in action execution
- Distributed multiagent reasoning
- Heuristics

WEBSITE `http://www.dimi.uniud.it/dovier/CLPASP`

Grazie Michael