DISSERTATION:

Advances in Answer Set Planning

Dipl.-Ing. Axel Polleres

axel@kr.tuwien.ac.at

supervised by

O.Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Eiter

Institut für Informationssysteme, Abteilung für Wissensbasierte Systeme

Overview – Contributions

- Preliminaries

Overview – Contributions

- Preliminaries
- Novel Declarative Planning Language $\mathcal{K}^c$
  - Syntax,Semantics
  - Complexity of Planning in $\mathcal{K}^c$

Overview – Contributions

- Preliminaries
- Novel Declarative Planning Language $\mathcal{K}^c$
  - Syntax,Semantics
  - Complexity of Planning in $\mathcal{K}^c$
- Problem Solving in Answer Set Programming (ASP) by "Guess and Check"

Overview – Contributions

- Preliminaries
- Novel Declarative Planning Language $\mathcal{K}^c$
  - Syntax,Semantics
  - Complexity of Planning in $\mathcal{K}^c$
- Problem Solving in Answer Set Programming (ASP) by "Guess and Check"
- Translation of Planning in $\mathcal{K}^c$ to ASP

Overview – Contributions

- Preliminaries
- Novel Declarative Planning Language $\mathcal{K}^c$
  - Syntax,Semantics
  - Complexity of Planning in $\mathcal{K}^c$
- Problem Solving in Answer Set Programming (ASP) by "Guess and Check"
- Translation of Planning in $\mathcal{K}^c$ to ASP
- Ready-to-Use Planning System $\texttt{DLV}^{\mathcal{K}}$
  - Implementation
  - Experiments

Overview – Contributions

- Preliminaries
- Novel Declarative Planning Language $\mathcal{K}^c$
  - Syntax,Semantics
  - Complexity of Planning in $\mathcal{K}^c$
- Problem Solving in Answer Set Programming (ASP) by "Guess and Check"
- Translation of Planning in $\mathcal{K}^c$ to ASP
- Ready-to-Use Planning System $\texttt{DLV}^{\mathcal{K}}$
  - Implementation
  - Experiments
- Application: Planning for MAS-Monitoring

Preliminaries:

- Planning Problem: Find a sequence of actions to bring an agent from an initial state to a goal state

Input:	A set of actions (preconditions, effects);
	Fluents (state variables) and their
	initial values and goal values
Output:	Sequence of action sets (discrete notion of time)

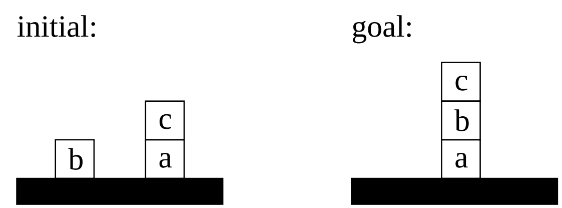
Preliminaries:

- Planning Problem: Find a sequence of actions to bring an agent from an initial state to a goal state

Input:	A set of actions (preconditions, effects);
	Fluents (state variables) and their
	initial values and goal values
Output:	Sequence of action sets (discrete notion of time)

- Classical Planning (complete knowledge, deterministic actions)

Actions: `move(B,L)`
Fluents: `on(b,table), on(c,a), on(a,table), occupied(a), ...`



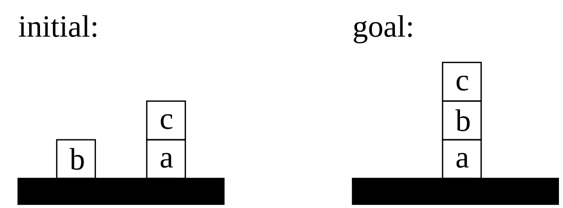
Preliminaries:

- Planning Problem: Find a sequence of actions to bring an agent from an initial state to a goal state

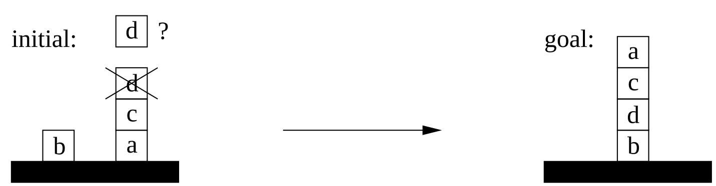
Input:	A set of actions (preconditions, effects);
	Fluents (state variables) and their
	initial values and goal values
Output:	Sequence of action sets (discrete notion of time)

- Classical Planning (complete knowledge, deterministic actions)

Actions: `move(B,L)`
Fluents: `on(b,table), on(c,a), on(a,table), occupied(a), ...`



- Non-Classical Planning(Conformant Plans, Conditional Plans, . . .)



Planning and Action Languages

Existing formal Languages: STRIPS, ADL, PDDL, $\mathcal{A}$, $\mathcal{C}$, . . .

Here: Novel planning language $\mathcal{K}^c$:

$\mathcal{K}^c$– Features:

- A relative of action languages $\mathcal{A}$ (Gelfond & Lifschitz, 1993) and $\mathcal{C}$ (Giunchiglia & Lifschitz, 1998)

Planning and Action Languages

Existing formal Languages: STRIPS, ADL, PDDL, $\mathcal{A}$, $\mathcal{C}$, . . .

Here: Novel planning language $\mathcal{K}^c$:

$\mathcal{K}^c$– Features:

- A relative of action languages $\mathcal{A}$ (Gelfond & Lifschitz, 1993) and $\mathcal{C}$ (Giunchiglia & Lifschitz, 1998)
- Incomplete states (“knowledge states”)

Planning and Action Languages

Existing formal Languages: STRIPS, ADL, PDDL, $\mathcal{A}$, $\mathcal{C}$, ...

Here: Novel planning language $\mathcal{K}^c$:

$\mathcal{K}^c$– Features:

- A relative of action languages $\mathcal{A}$ (Gelfond & Lifschitz, 1993) and $\mathcal{C}$ (Giunchiglia & Lifschitz, 1998)
- Incomplete states ("knowledge states")
- Default (nonmonotonic) negation and strong (classical) negation

Planning and Action Languages

Existing formal Languages: STRIPS, ADL, PDDL, $\mathcal{A}$, $\mathcal{C}$, ...

Here: Novel planning language $\mathcal{K}^c$:

$\mathcal{K}^c$– Features:

- A relative of action languages $\mathcal{A}$ (Gelfond & Lifschitz, 1993) and $\mathcal{C}$ (Giunchiglia & Lifschitz, 1998)
- Incomplete states (“knowledge states”)
- Default (nonmonotonic) negation and strong (classical) negation
- Nondeterministic action effects

Planning and Action Languages

Existing formal Languages: STRIPS, ADL, PDDL, $\mathcal{A}$, $\mathcal{C}$, ...

Here: Novel planning language $\mathcal{K}^c$:

$\mathcal{K}^c$– Features:

- A relative of action languages $\mathcal{A}$ (Gelfond & Lifschitz, 1993) and $\mathcal{C}$ (Giunchiglia & Lifschitz, 1998)
- Incomplete states ("knowledge states")
- Default (nonmonotonic) negation and strong (classical) negation
- Nondeterministic action effects
- Action costs

$\mathcal{K}^c$ Planning Domains and Problems

Background Knowledge Π: A logic program Π with a single model, (answer set) defining type information and static knowledge.

$\mathcal{K}$ Action Description AD:

fluents: D_F % fluent declarations

actions: D_A % action type declarations

always: C_R % causation rules + exec. cond's

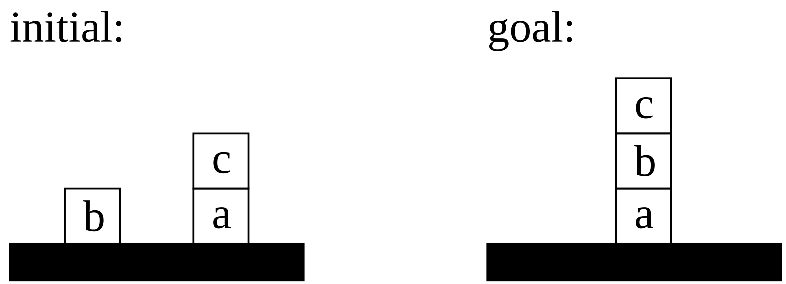
initially: C_I % initial state constraints

$\mathcal{K}$ Planning Domain: $\langle \Pi, AD \rangle$

$\mathcal{K}$ Planning Problem: additional goal

goal: $G?(i)$ ground literal(s) G; plan length $i \geq 0$.

Blocks world in $\mathcal{K}^c$



Background knowledge

```
Π = { block(a). block(b). block(c).
      location(table).
      location(L) :- block(L). }
```

(Logic Program which has a single model - set of “invariant” facts)

Blocks world: $\mathcal{K}^c$ Problem description

```
fluents:    on(B,L) requires block(B), location(L).
            occupied(B) requires location(B).
actions:    move(B,L) requires block(B), location(L) costs 1.
```

Blocks world: $\mathcal{K}^c$ Problem description

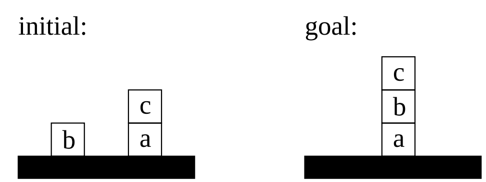
```
fluents:    on(B,L) requires block(B), location(L).
            occupied(B) requires location(B).
actions:    move(B,L) requires block(B), location(L) costs 1.

always:    executable move(B,L) if not occupied(B), not occupied(L), B<>L.
           caused on(B,L) after move(B,L).
           caused ¬on(B,L1) after move(B,L), on(B,L1), L<>L1.
           caused occupied(B) if on(B1,B), block(B).
           inertial on(B,L). % Explicit frame axioms!
           noConcurrency. % Optionally, parallel actions!
```

Blocks world: $\mathcal{K}^c$ Problem description

```
fluents:    on(B,L) requires block(B), location(L).
            occupied(B) requires location(B).
actions:    move(B,L) requires block(B), location(L) costs 1.

always:    executable move(B,L) if not occupied(B), not occupied(L), B<>L.
           caused on(B,L) after move(B,L).
           caused ¬on(B,L1) after move(B,L), on(B,L1), L<>L1.
           caused occupied(B) if on(B1,B), block(B).
           inertial on(B,L). % Explicit frame axioms!
           noConcurrency. % Optionally, parallel actions!

initially: on(a,table). on(b,table). on(c,a).

goal: on(c,b),on(b,a),on(a,table)? (3)
```

Blocks world: $\mathcal{K}^c$ Problem description

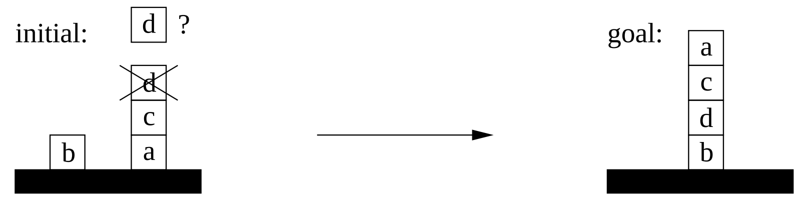
```
fluents:     on(B,L) requires block(B), location(L).
             occupied(B) requires location(B).
actions:     move(B,L) requires block(B), location(L) costs 1.

always:     executable move(B,L) if not occupied(B), not occupied(L), B<>L.
            caused on(B,L) after move(B,L).
            caused ¬on(B,L1) after move(B,L), on(B,L1), L<>L1.
            caused occupied(B) if on(B1,B), block(B).
            inertial on(B,L). % Explicit frame axioms!
            noConcurrency. % Optionally, parallel actions!

initially: on(a,table). on(b,table). on(c,a).

goal: on(c,b),on(b,a),on(a,table)? (3)
```



Intuitively: Feasible plan is

```
move(c,table); move(b,a); move(c,b) COSTS: 3
```

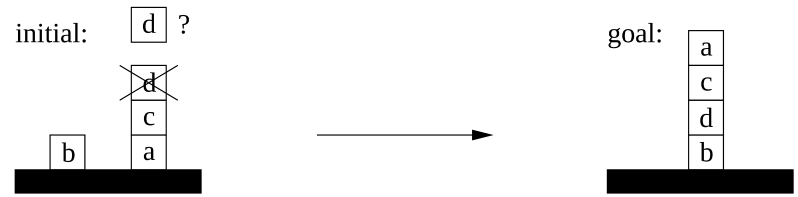
Blocks World in $\mathcal{K}^c$ (cont'd)



Incomplete Knowldege:

```
initially: total on(d,L).
           caused ¬on(d,c).
           % State Axioms:
           caused false if on(B,L), on(B,L1), L<>L1.
           caused false if on(B1,B), on(B2,B), block(B), B1<>B2.
           ⋮
goal: on(a,c),on(c,d),on(d,b),on(b,table)? (4)
```

Blocks World in $\mathcal{K}^c$ (cont'd)



Incomplete Knowldege:

```
initially: total on(d,L).
           caused ¬on(d,c).
           % State Axioms:
           caused false if on(B,L), on(B,L1), L<>L1.
           caused false if on(B1,B), on(B2,B), block(B), B1<>B2.
           ⋮
goal: on(a,c),on(c,d),on(d,b),on(b,table)? (4)
```

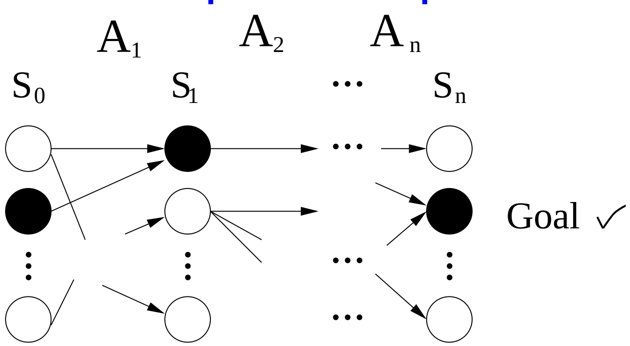
Feasible plans:

```
move(c,d); move(a,c); (no action); (no action) COSTS: 2
move(d,c); move(d,b); move(c,d); move(a,c) COSTS: 4
```

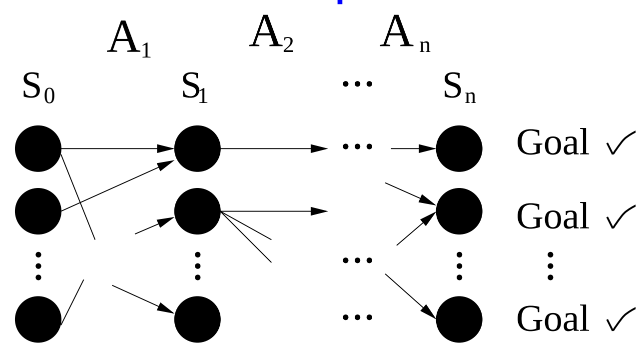
Semantics of $\mathcal{K}^c$ – Plans

Multi-valued transition function $t(s, A)$, LP-based (Answer Sets!)

Optimistic plans:



Secure plans:



Optimal plans: plans with lowest cost

Admissible plans: plans which stay within fixed cost limit

$\mathcal{K}^c$ Complexity

PD	plan length i in query $q = Goal\ ?\ (i)$	
	fixed (=constant)	arbitrary
general	NP / Π_2^P / Σ_3^P -complete	PSPACE / Π_2^P / NEXPTIME -complete
proper	NP / co-NP / Σ_2^P -complete	PSPACE / co-NP / NEXPTIME -complete

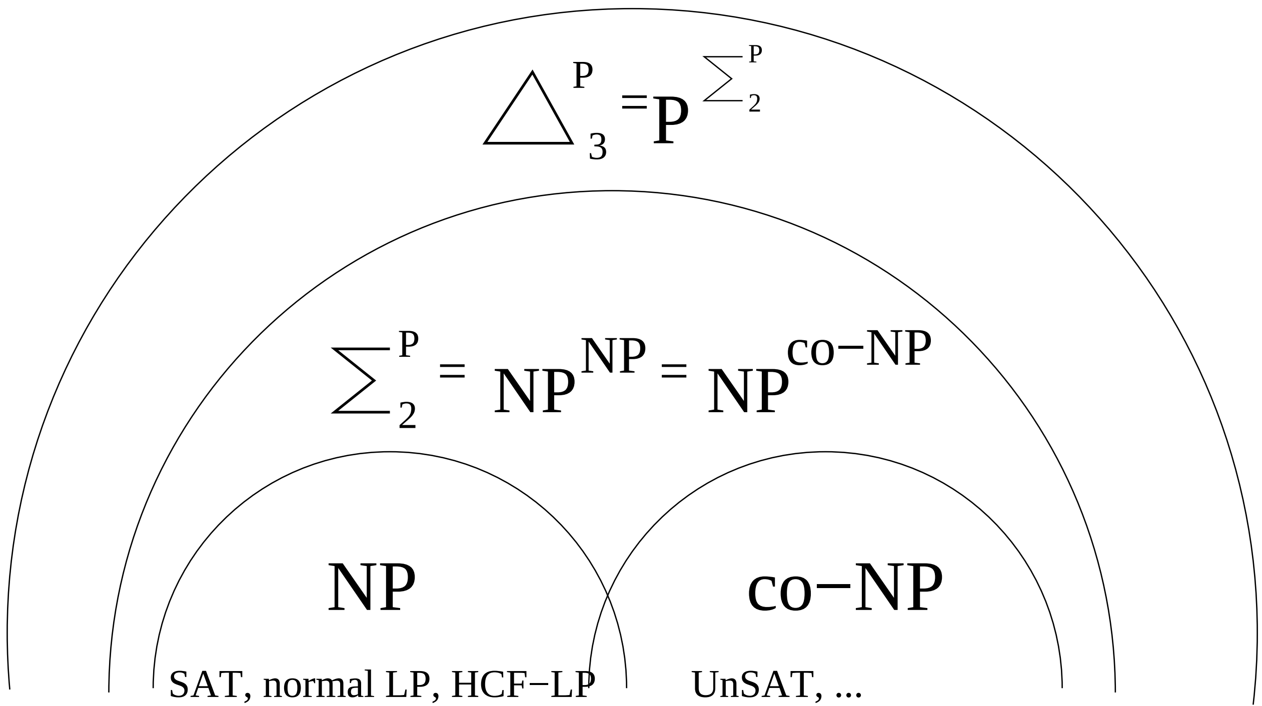
Complexity Results for Optimistic Planning / Security Checking / Secure Planning in $\mathcal{K}$ (Propositional Case)

Planning with Action costs:
Computing optimal optimistic/secure plans is $\mathrm{F}\Delta_2^P$-complete/$\mathrm{F}\Delta_4^P$-complete.

Answer Set Programming (with weak constraints) can be used to solve some of these tasks!

Answer Set Programming (ASP)

ASP with weak constraints is capable of solving problems beyond NP!
(conformant planning for proper domains, general secure checking)



- Solvers for NP problems: Smodels, SAT-Solvers, . . .
- Solvers for Σ_2^P problems: DLV, GnT, . . .
- Solvers for Δ_3^P problems: DLV with weak constraints . . .

Answer Set Programming (ASP)

- function-free, disjunctive Logic Programs, set of rules:

$$h_1 \texttt{ v } \ldots \texttt{ v } h_l \texttt{ :- } b_1, \ldots, b_m, \texttt{not } b_{m+1}, \ldots \texttt{not } b_n.$$

- Semantics: Answer Sets Semantics for nonmonotonic logic programs (Gelfond & Lifschitz, 1991), minimal "stable" models

- Extension: weak constraints (Buccafurri et.al., 1999):

$$:\sim \; b_1, \ldots, b_m, \texttt{not } b_{m+1}, \ldots \texttt{not } b_n.[C]$$

- Semantics: Optimal Answer Sets (with minimal violation costs)

Problem Solving in ASP:

- "Guess and Check" Paradigm: a Simple Example:

```
col(X, r) v col(X, g) v col(X, b) :- node(X).   } Guess
:- edge(X, Y), col(X, C), col(Y, C).            } Solution Check
```

Input: A graph represented by `node(_)` and `edge(_, _)`.

Problem: Assign a color to all nodes such that adjacent nodes always have different colors.

NP-complete problem!

ASP is well suited for solving search problems with a finite search space! Efficient solvers (DLV, smodels, . . .) exist!

Beyond NP: 2QBFs

$$\Psi = \exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \psi$$

$\psi = d_1 \vee \cdots \vee d_k$

$d_i = a_{i,1} \wedge \cdots \wedge a_{i,l_i}$ and $|a_{i,j}| \in \{x_1, \ldots, x_m, y_1, \ldots, y_n\}$

Compute an assignment to $x_1, \ldots, x_m$ such that Ψ is true

Beyond NP: 2QBFs

$$\Psi = \exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \psi$$

$\psi = d_1 \vee \cdots \vee d_k$

$d_i = a_{i,1} \wedge \cdots \wedge a_{i,l_i}$ and $|a_{i,j}| \in \{x_1, \ldots, x_m, y_1, \ldots, y_n\}$

Compute an assignment to $x_1, \ldots, x_m$ such that Ψ is true

```
x1 v nx1. ...xm v nxm.                        } Guess

y1 v ny1. ...yn v nyn.                        }
sat :- a1,1,...,a1,l1 .                       }
      ⋮                                       }
sat :- ak,1,...,ak,lk .                       } Check
y1 :- sat.  ... yn :- sat.                    }
ny1 :- sat.  ... nyn :- sat.                  }
:- not sat.                                   }
```

Check part uses “saturation” technique!

Integrate “Guess” and “Check”

Problems:

- Integrated Σ_2^P programs often hard to find,
- “Guess” and “Check” structure hard to see.
- Σ_2^P encodings use unintuitive “saturation” techniques

Integrate "Guess" and "Check"

Problems:

- Integrated Σ_2^P programs often hard to find,
- "Guess" and "Check" structure hard to see.
- Σ_2^P encodings use unintuitive "saturation" techniques

Solution: Interleaved Computation!

- Separate programs Π_{guess} and Π_{check}
- Compute solutions interleaved, i.e. compute all solutions S to Π_{guess} and only accept those where $\Pi_{check}(S)$ has no answer set.

Integrate "Guess" and "Check"

Problems:

- Integrated Σ_2^P programs often hard to find,
- "Guess" and "Check" structure hard to see.
- Σ_2^P encodings use unintuitive "saturation" techniques

Solution: Interleaved Computation!

- Separate programs Π_{guess} and Π_{check}
- Compute solutions interleaved, i.e. compute all solutions S to Π_{guess} and only accept those where $\Pi_{check}(S)$ has no answer set.
- Thesis provides **automatic method** for combining these programs!

ASP Translation for Planning Problems

Based on this method, we define ASP Translations for:

- optimistic planning
- general/proper secure checking
- proper secure planning

ASP Translation for Planning Problems

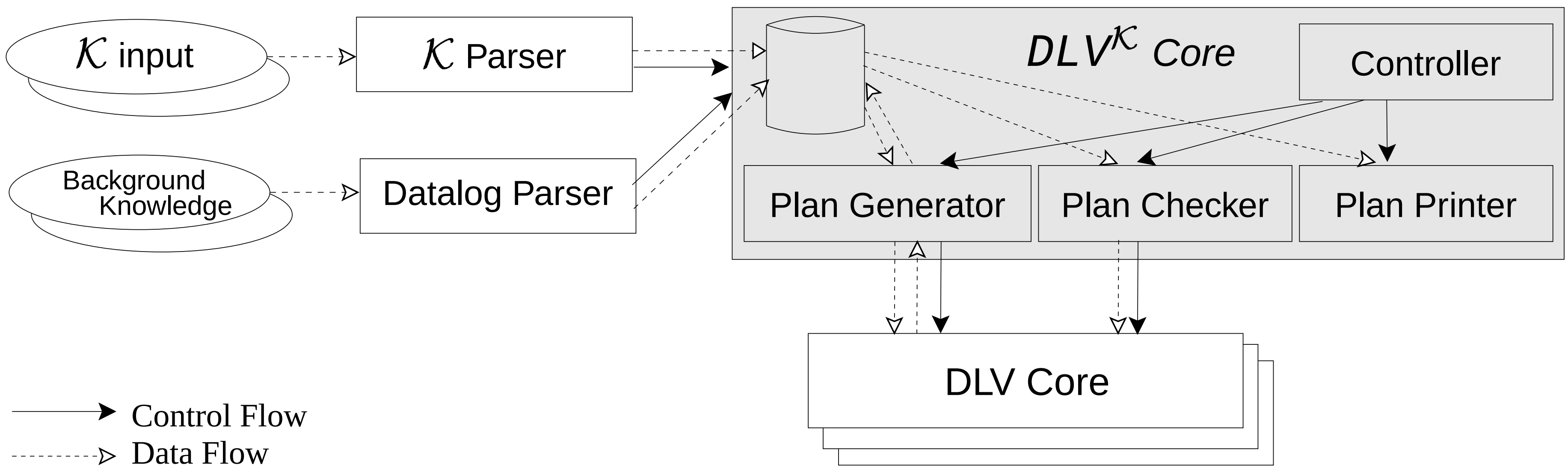
Based on this method, we define ASP Translations for:

- optimistic planning
- general/proper secure checking
- proper secure planning

Action costs (optimal/admissible planning):

- Extend these translations by weak constraints feature

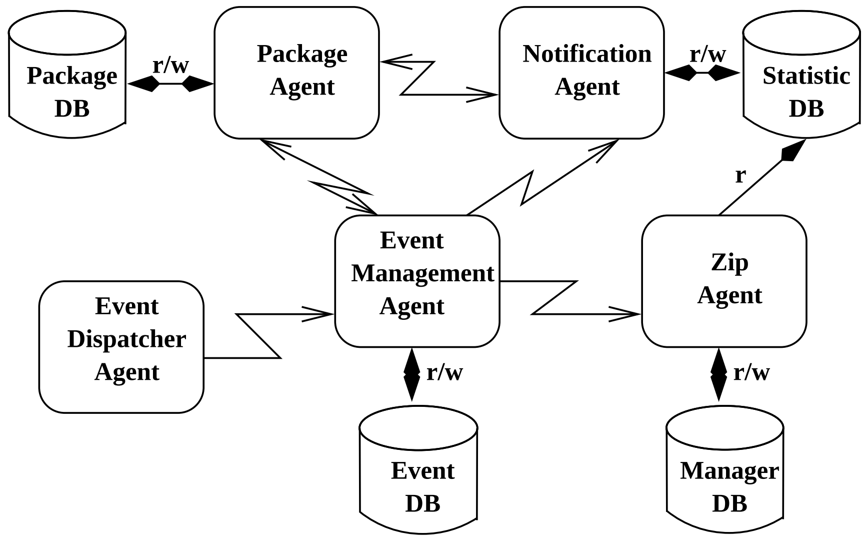
Implementation – $DLV^{\mathcal{K}}$



Plan Generator: Computes Optimistic Plans

Plan Checker: Checks Optimistic Plans for Security

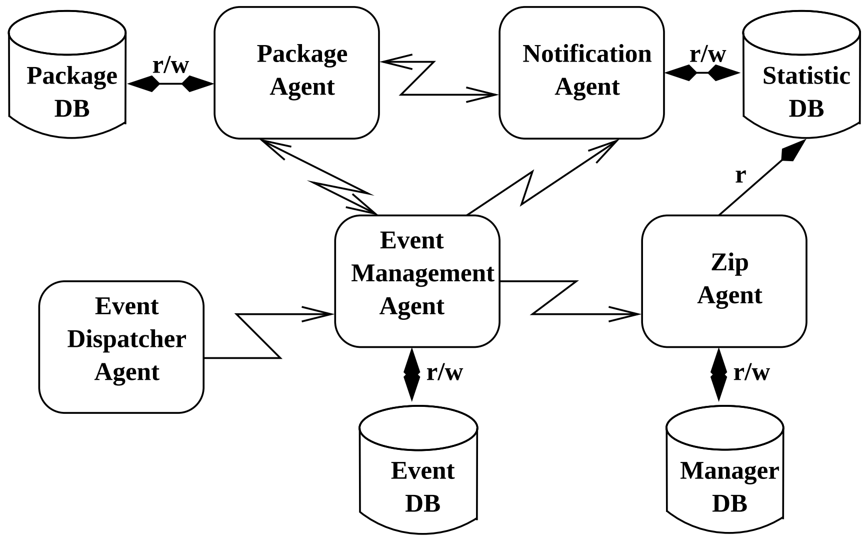
Planning for Multi-Agent Monitoring



Idea:

- Model collaborative Behavior in an MAS in $\mathcal{K}^c$, and
- derive valid messaging protocols from plans.

Planning for Multi-Agent Monitoring



Idea:

- Model collaborative Behavior in an MAS in $\mathcal{K}^c$, and
- derive valid messaging protocols from plans.

Further interesting applications:

- Optimal Route planning with exceptional, time-dependent costs
- Cheapest among the shortest plans, Shortest among the cheapest plans
- Conformant planning examples from the literature (SQUARE, Bomb in Toilet)

Conclusions & Outlook

- Expressive planning language based on Logic Programming (Knowledge Representation!)

Conclusions & Outlook

- Expressive planning language based on Logic Programming (Knowledge Representation!)
- Efficient planning system based on Answer Set Techniques is feasible: $\texttt{DLV}^{\mathcal{K}}$

Conclusions & Outlook

- Expressive planning language based on Logic Programming (Knowledge Representation!)
- Efficient planning system based on Answer Set Techniques is feasible: $\mathtt{DLV}^{\mathcal{K}}$
- Encouraging Results (Experiments!)

Conclusions & Outlook

- Expressive planning language based on Logic Programming (Knowledge Representation!)
- Efficient planning system based on Answer Set Techniques is feasible: $\texttt{DLV}^{\mathcal{K}}$
- Encouraging Results (Experiments!)
- Exploiting the full potential of ASP power (Σ_2^P encodings), problems beyond SAT Solvers!

Conclusions & Outlook

- Expressive planning language based on Logic Programming (Knowledge Representation!)
- Efficient planning system based on Answer Set Techniques is feasible: $\texttt{DLV}^{\mathcal{K}}$
- Encouraging Results (Experiments!)
- Exploiting the full potential of ASP power (Σ_2^P encodings), problems beyond SAT Solvers!
- Further Work: Reactive Planning!

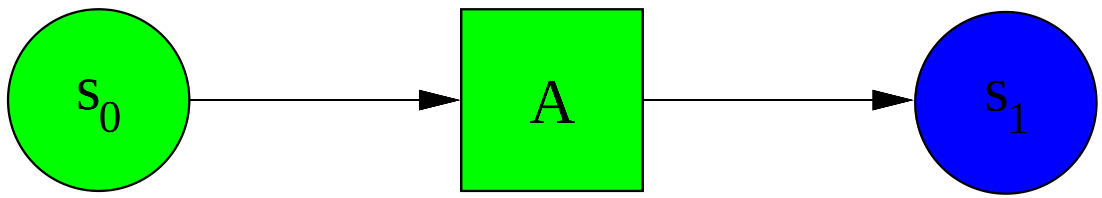
Selected Publications

- T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. *A Logic Programming Approach to Knowledge-State Planning: Semantics and Complexity*. ACM Transactions on Computational Logic, 2003. To appear.
- T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. *A Logic Programming Approach to Knowledge-State Planning, II: the* $\mathtt{DLV}^{\mathcal{K}}$ *System*. Artificial Intelligence, 144(1–2):157–211, March 2003.
- T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. *Answer Set Planning under Action Costs*. Journal of Artificial Intelligence Research, 19:25–71, 2003.
- J. Dix, T. Eiter, M. Fink, A. Polleres, and Y. Zhang. *Monitoring agents using planning.* German Conference on Artificial Intelligence (KI2003), 2003.
- T. Eiter and A. Polleres. *Towards Automated Integration of Guess and Check Programs in Answer Set Programming*. Accepted for 7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-7).

Semantics of $\mathcal{K}^c$ – Transitions

Transition-based semantics: "legal" transitions $\langle s_0, A, s_1 \rangle$

caused fl if Cond1 after Cond2



A . . . set of actions (executable)

- Cond2 is evaluated in s_0
- fl and Cond1 are evaluated in s_1

Define new state s_1 by a non-monotonic logic program of rules

fl :- Cond1

Remark:

e.g., transitive closure easily expressed (LP-flavored semantics of $\mathcal{K}$)