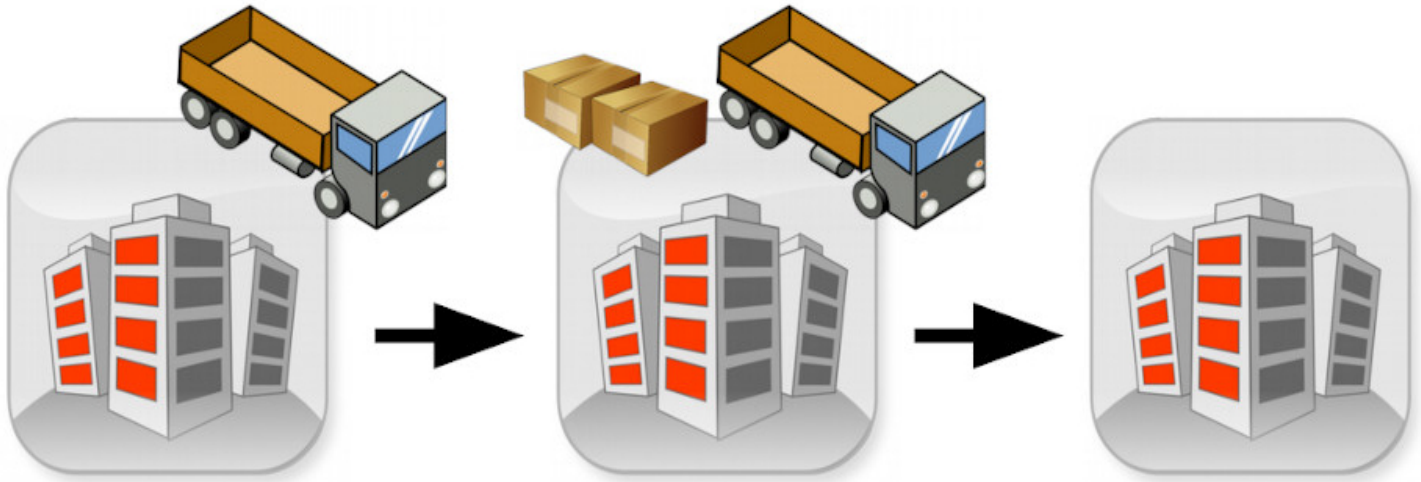
Karlsruhe Institute of Technology

PASAR - Planning as Satisfiability with Abstraction Refinement

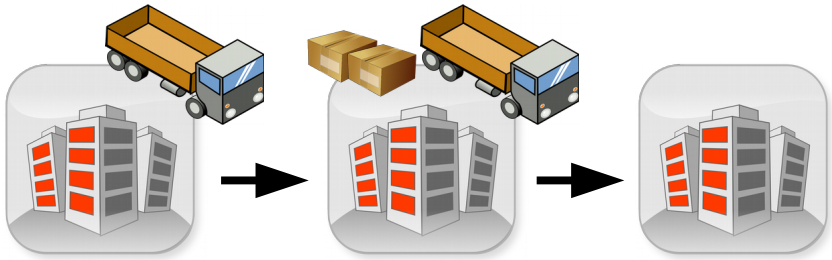
12th Annual Symposium on Combinatorial Search

Nils Froleyks (Speaker), Tomáš Balyo, Dominik Schreiber | July 17, 2019

INSTITUT FÜR THEORETISCHE INFORMATIK

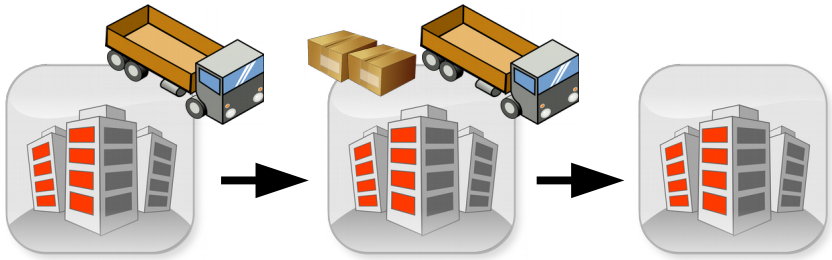


Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

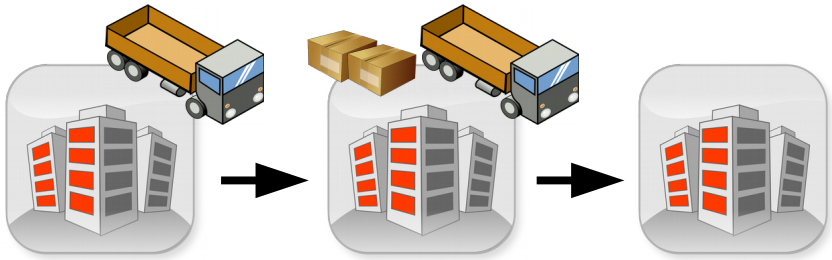
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- (World) State: Consistent set of boolean atoms:
  $\texttt{at}(T_1, A)$, $\texttt{at}(T_2, B)$, $\texttt{at}(P_1, B)$, $\texttt{at}(P_2, B)$,
  $\texttt{empty}(T_1)$, $\texttt{empty}(T_2)$, $\texttt{road}(A, B)$, $\texttt{road}(B, C)$

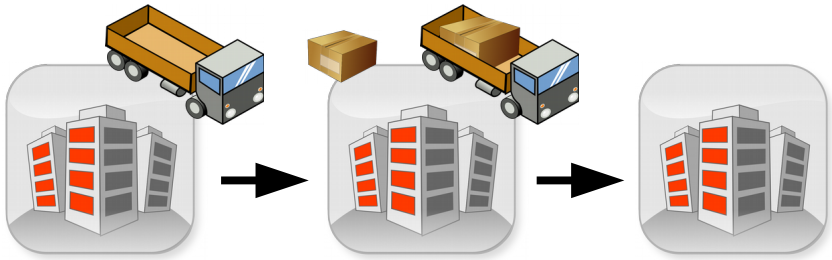
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- $\texttt{pickup}(T_2, P_1, B)$
  requires $\texttt{empty}(T_2), \texttt{at}(T_2, B), \texttt{at}(P_1, B)$
  deletes $\texttt{at}(P_1, B), \texttt{empty}(T_2)$ adds $\texttt{in}(P_1, T_2)$

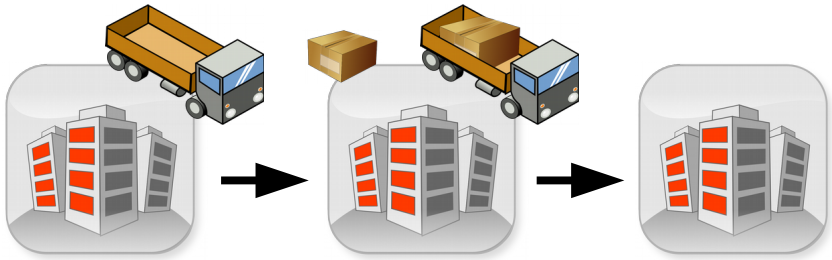
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- $\texttt{pickup}(T_2, P_1, B)$
  requires $\texttt{empty}(T_2), \texttt{at}(T_2, B), \texttt{at}(P_1, B)$
  deletes $\texttt{at}(P_1, B), \texttt{empty}(T_2)$ adds $\texttt{in}(P_1, T_2)$

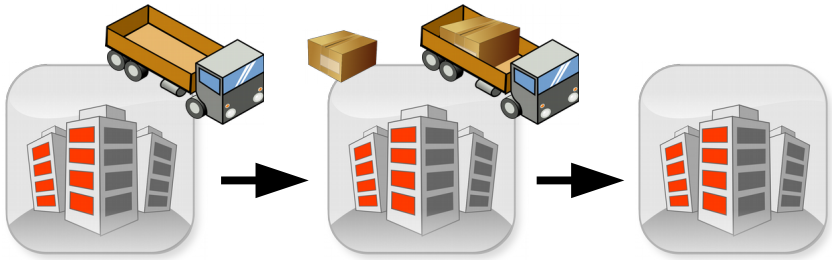
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- Goal g: Set of Atoms that must hold,
  e.g. $\texttt{at}(P_1, C)$ and $\texttt{at}(P_2, C)$

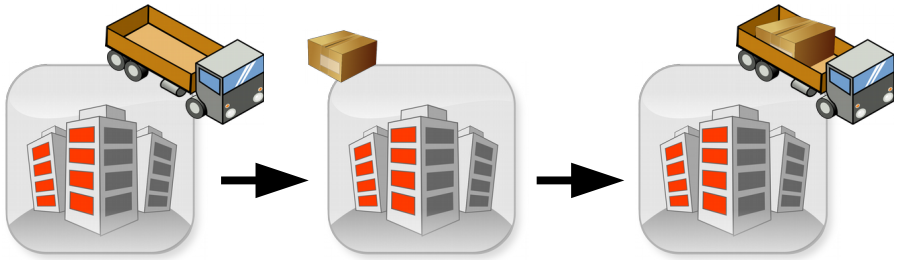
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- Plan $\pi = \langle \texttt{pickup}(T_2, P_1, B)$

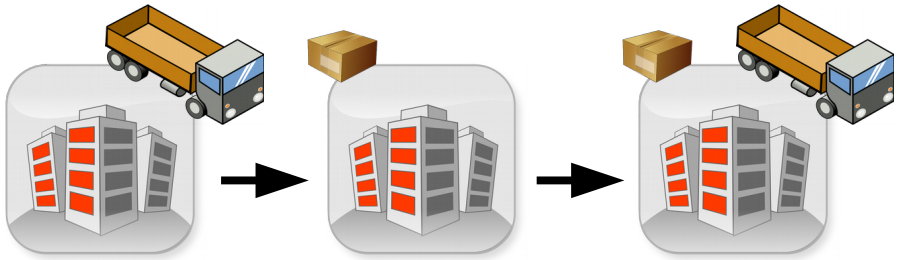
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- Plan $\pi = \langle \texttt{pickup}(T_2, P_1, B), \texttt{move}(T_2, B, C) \rangle$

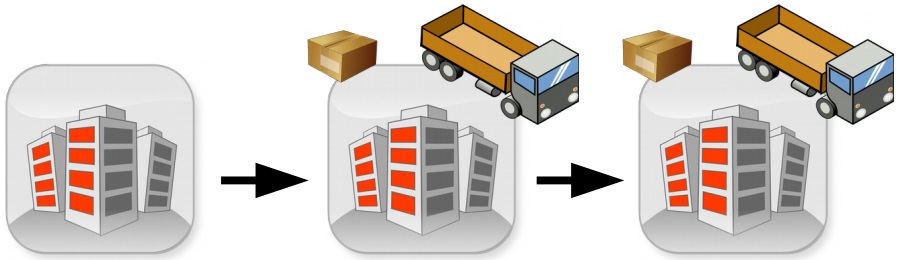
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- Plan $\pi = \langle \texttt{pickup}(T_2, P_1, B), \texttt{move}(T_2, B, C), \texttt{drop}(T_2, P_1, C)$

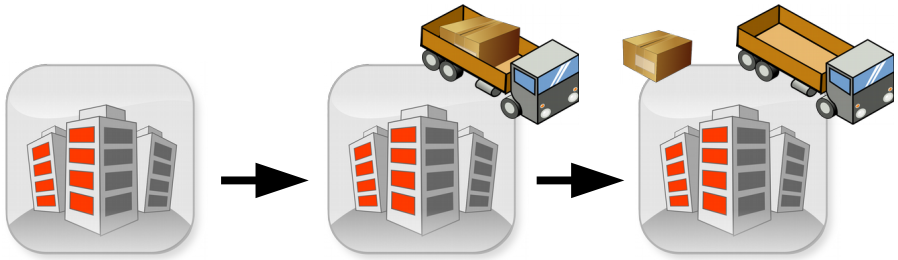
Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- Plan $\pi = \langle \texttt{pickup}(T_2, P_1, B), \texttt{move}(T_2, B, C), \texttt{drop}(T_2, P_1, C), \texttt{move}(T_1, A, B)$

Planning as Satisfiability with Abstraction Refinement



Find a valid sequence of actions from some initial world state to a desired goal state.

- Plan $\pi = \langle \texttt{pickup}(T_2, P_1, B), \texttt{move}(T_2, B, C), \texttt{drop}(T_2, P_1, C), \texttt{move}(T_1, A, B), \texttt{pickup}(T_1, P_2, B)$

Planning as Satisfiability with Abstraction Refinement

Find a valid sequence of actions from some initial world state to a desired goal state.

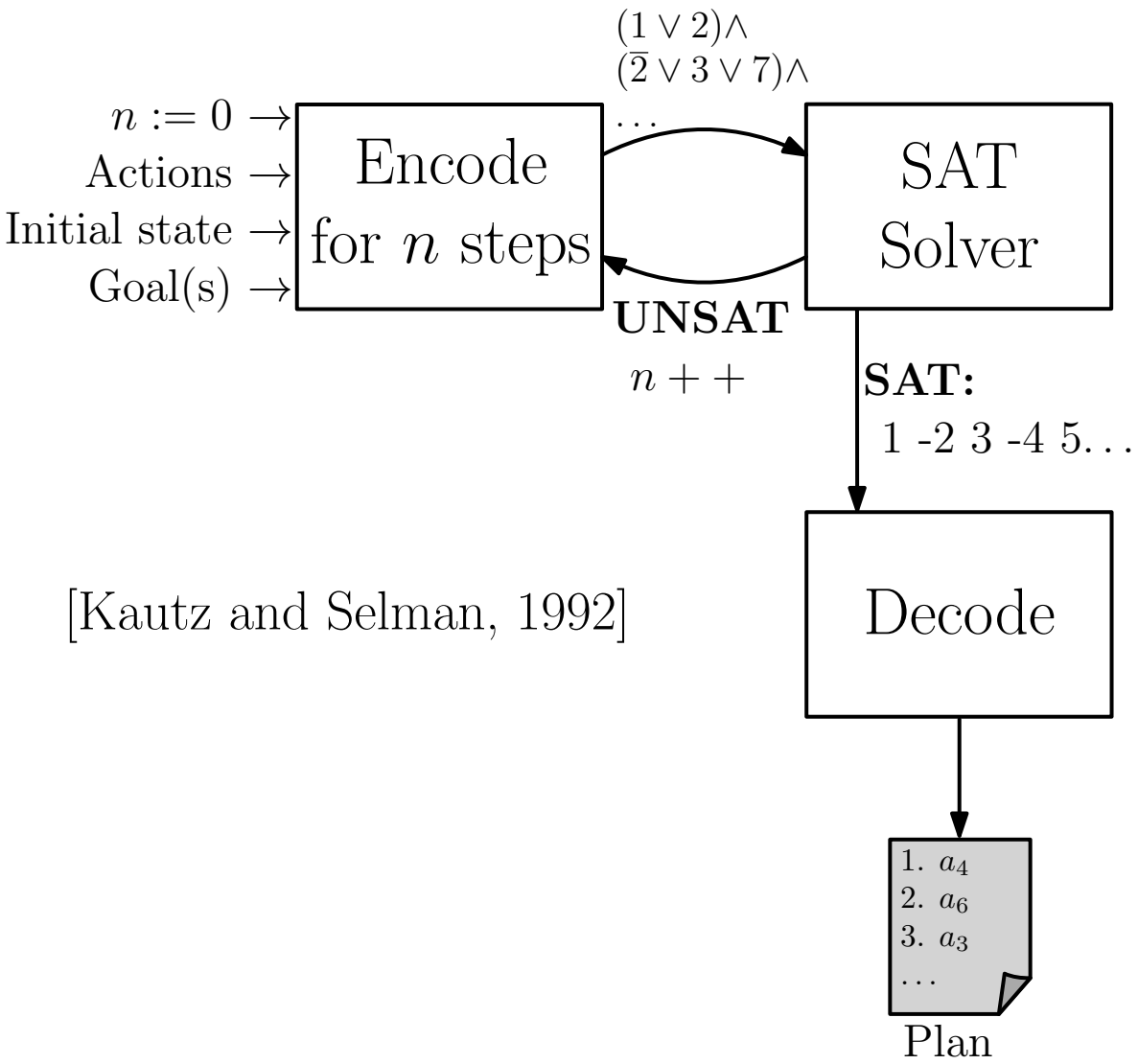
- Plan $\pi = \langle \texttt{pickup}(T_2, P_1, B), \texttt{move}(T_2, B, C), \texttt{drop}(T_2, P_1, C), \texttt{move}(T_1, A, B), \texttt{pickup}(T_1, P_2, B), \texttt{move}(T_1, B, C)$

Planning as Satisfiability with Abstraction Refinement

Find a valid sequence of actions from some initial world state to a desired goal state.

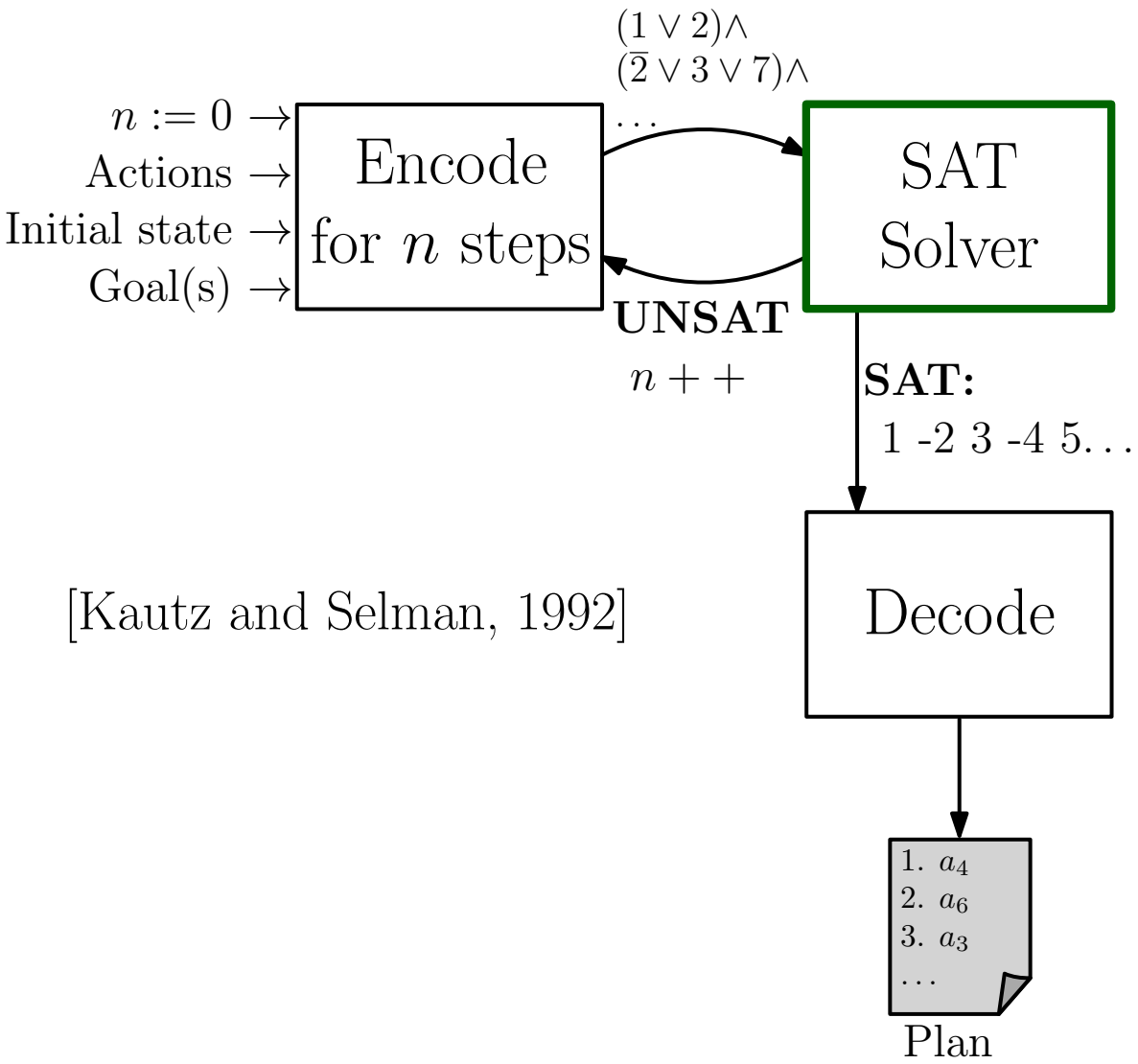
- Plan $\pi = \langle \texttt{pickup}(T_2, P_1, B), \texttt{move}(T_2, B, C), \texttt{drop}(T_2, P_1, C), \texttt{move}(T_1, A, B), \texttt{pickup}(T_1, P_2, B), \texttt{move}(T_1, B, C), \texttt{drop}(T_1, P_2, C) \rangle$

Planning as Satisfiability with Abstraction Refinement



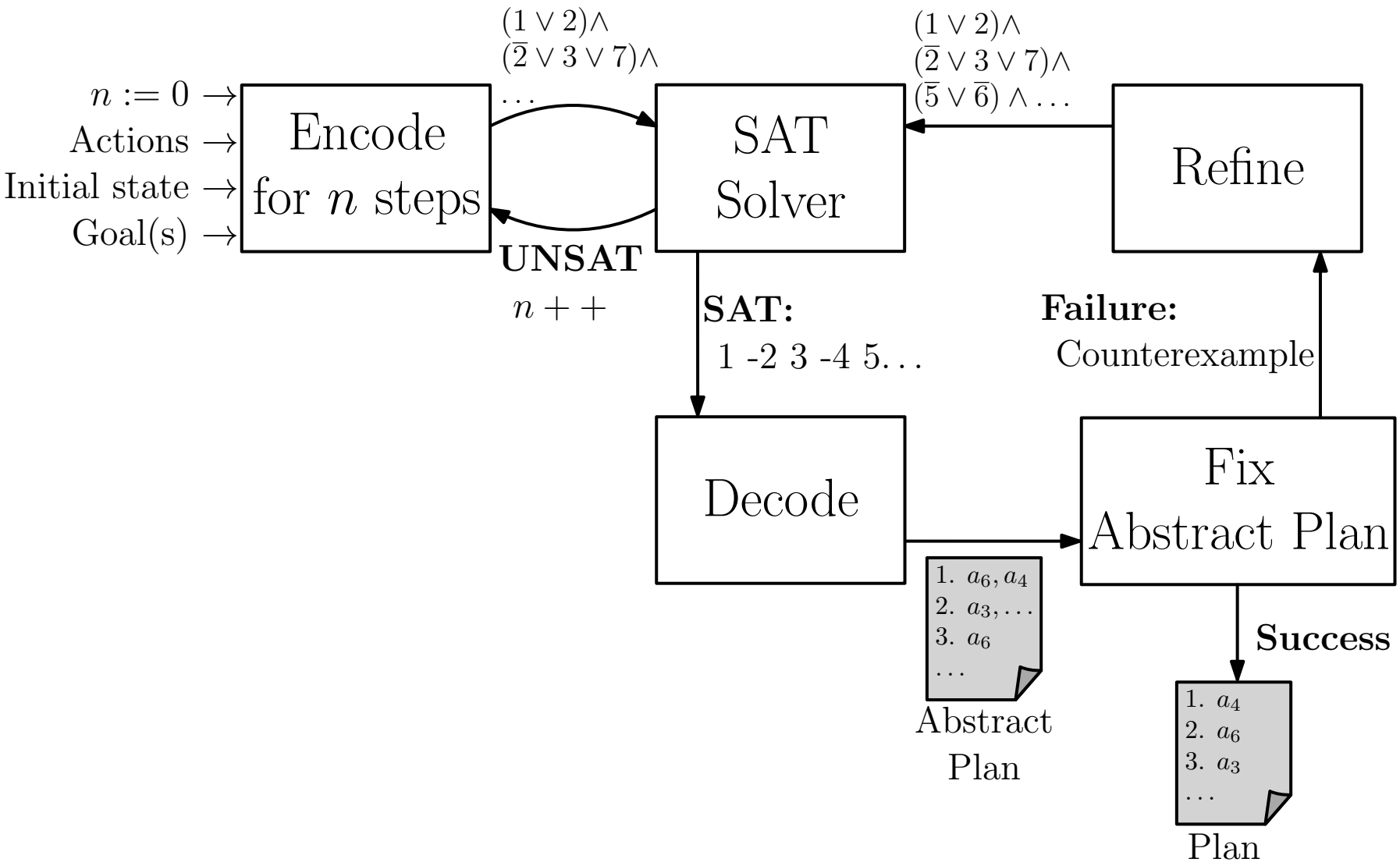
[Kautz and Selman, 1992]

Planning as Satisfiability with Abstraction Refinement

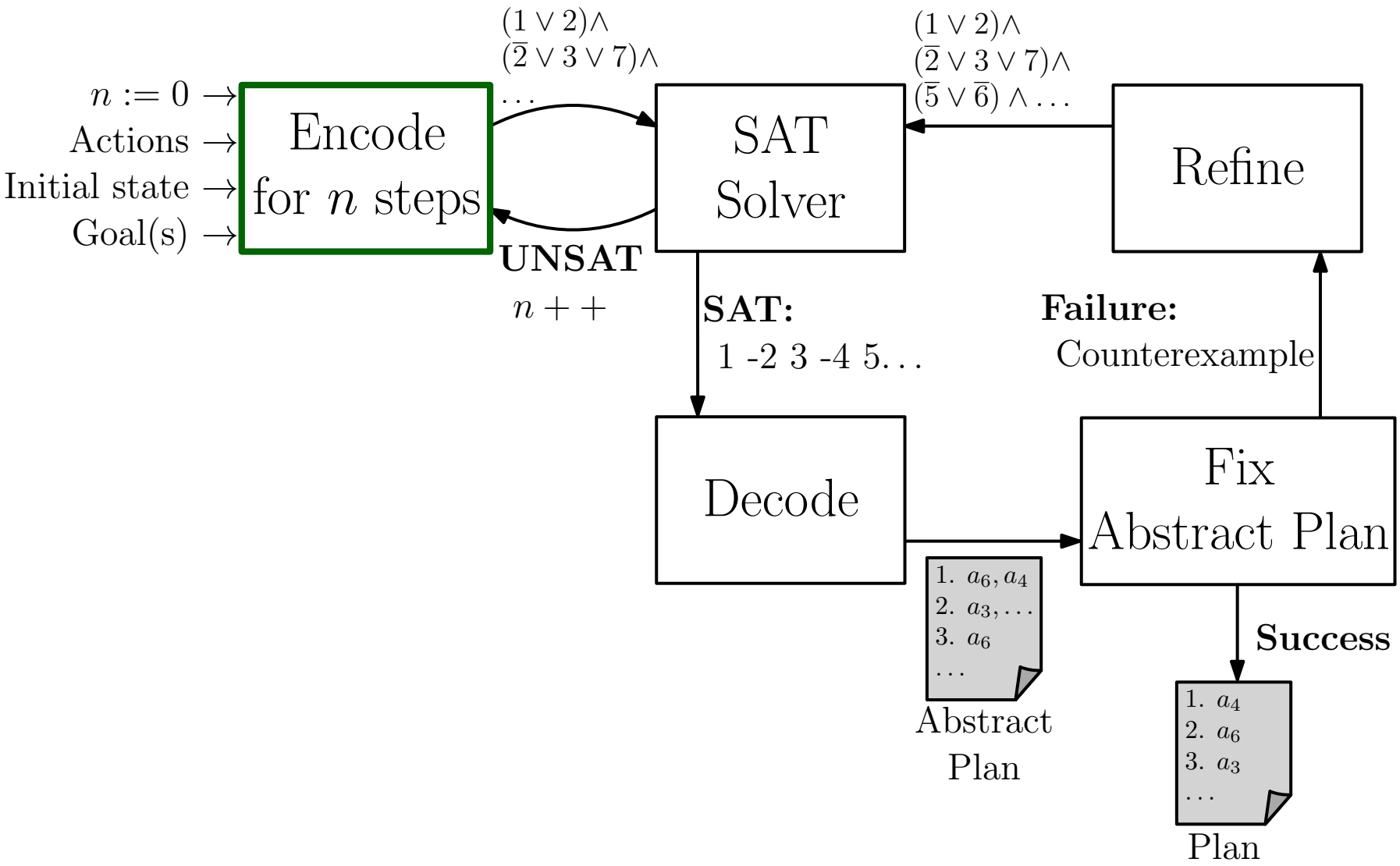


[Kautz and Selman, 1992]

Planning as Satisfiability with Abstraction Refinement



Planning as Satisfiability with Abstraction Refinement

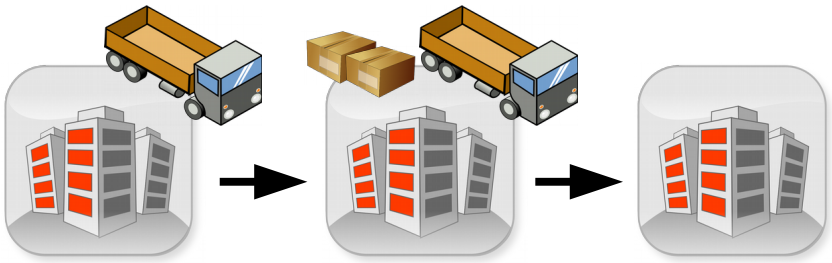


Encoding

State variable is_p^t and action variable do_a^t for each step $t = 0 \ldots n$

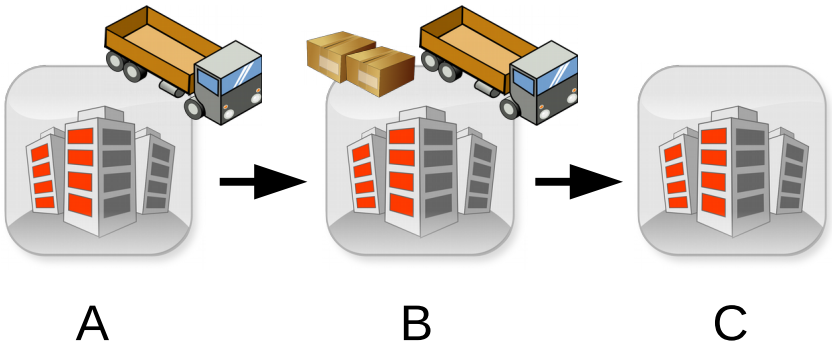
1. Initial state
2. Goal conditions
3. Preconditions are satisfied at step t
4. Effects hold at step $t + 1$
5. Frame axioms (state doesn't change arbitrarily)
6.

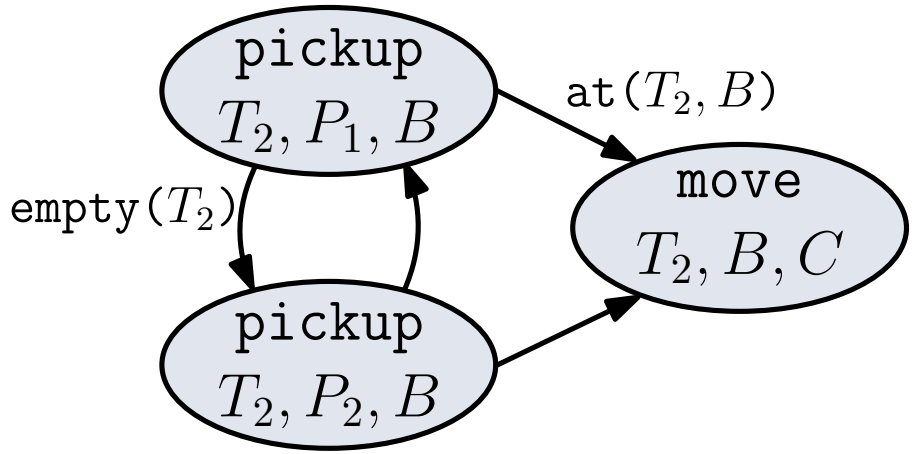
Interference



1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

Interference



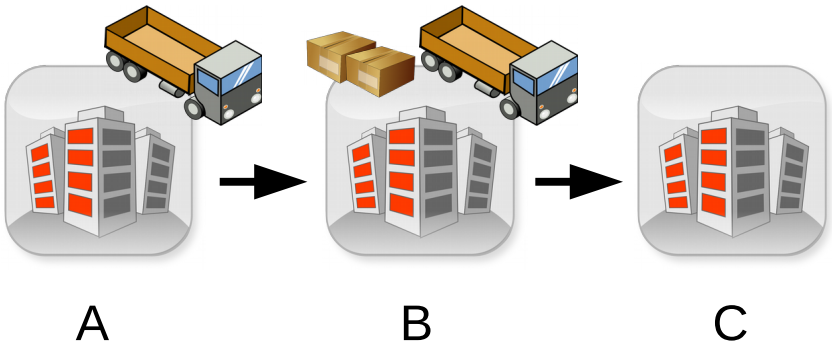


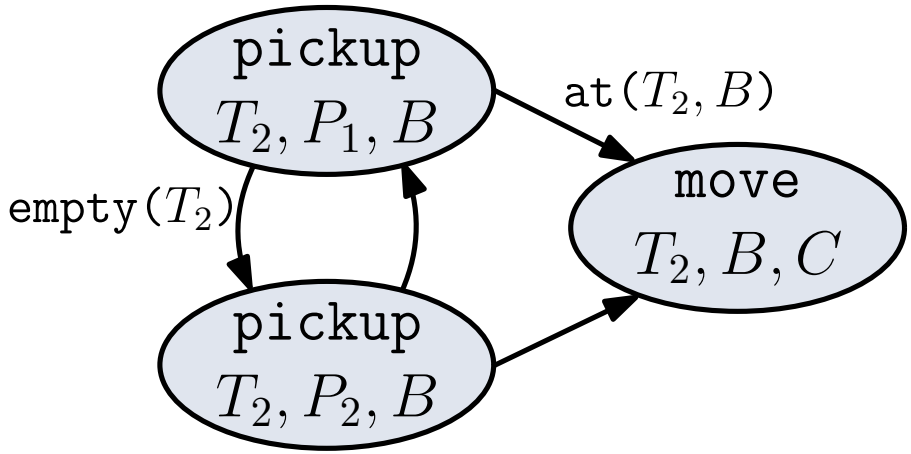
1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

Disabling graphs

- $V =$ Actions
- $(A, B) \in E \Leftrightarrow A$ requires a precondition that is deleted by B

Interference





1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

- $\forall$-step [Rintanen et al., 2006]
- Interference $(A, B) \Rightarrow \overline{A} \vee \overline{B}$

Encoding

State variable is_p^t and action variable do_a^t for each step $t = 0 \ldots n$

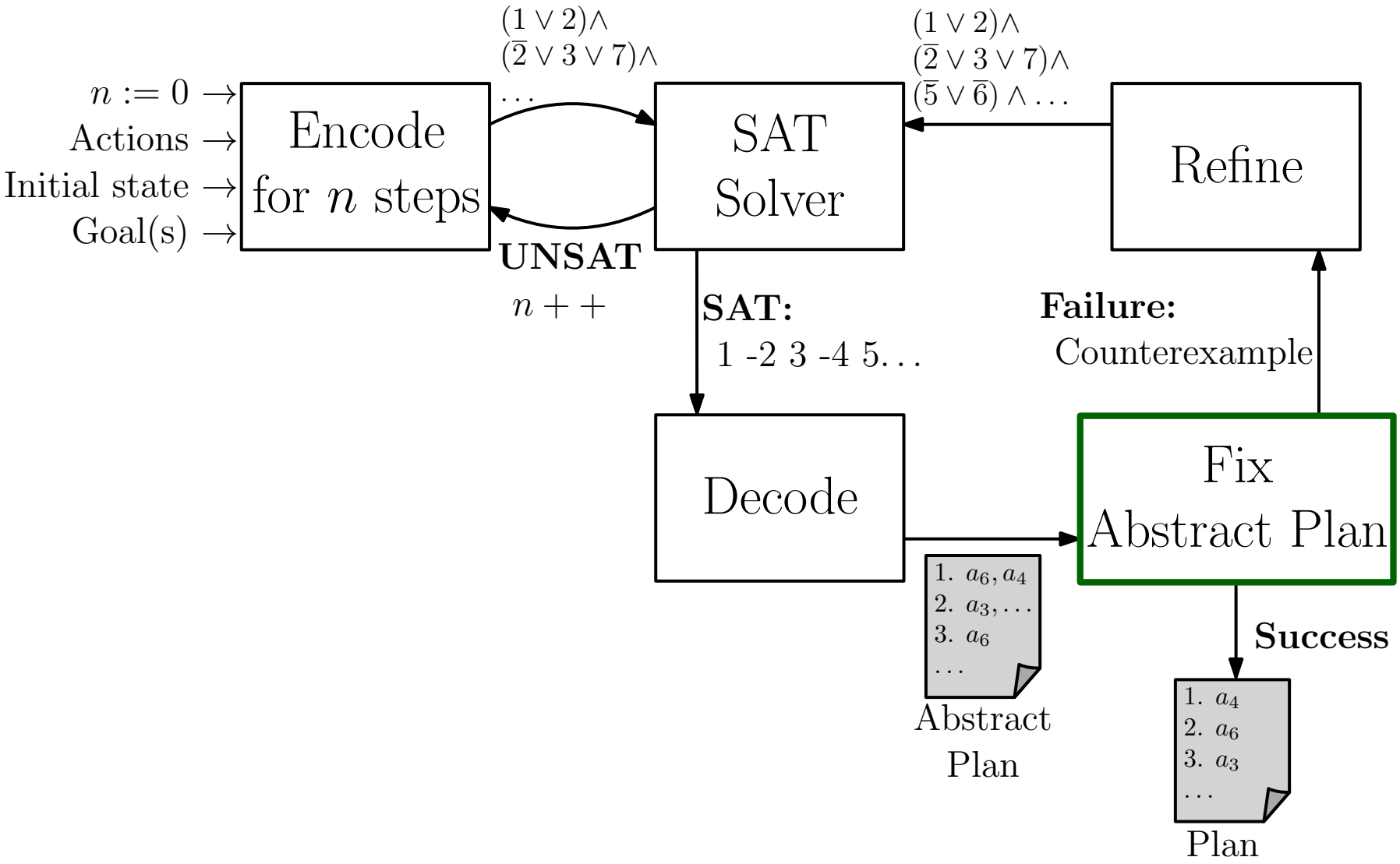
1. Initial state $\mathcal{O}(|P|)$
2. Goal conditions $\mathcal{O}(|P|)$
3. Preconditions are satisfied at step t $\mathcal{O}(n \cdot |A| \cdot |E|)$
4. Effects hold at step $t + 1$ $\mathcal{O}(n \cdot |A| \cdot |E|)$
5. Frame axioms (state doesn't change arbitrarily) $\mathcal{O}(n \cdot |P|)$
6. Interference $\mathcal{O}(n \cdot |A|^2)$
   - Sequential action execution or $\forall$-step encoding

Encoding

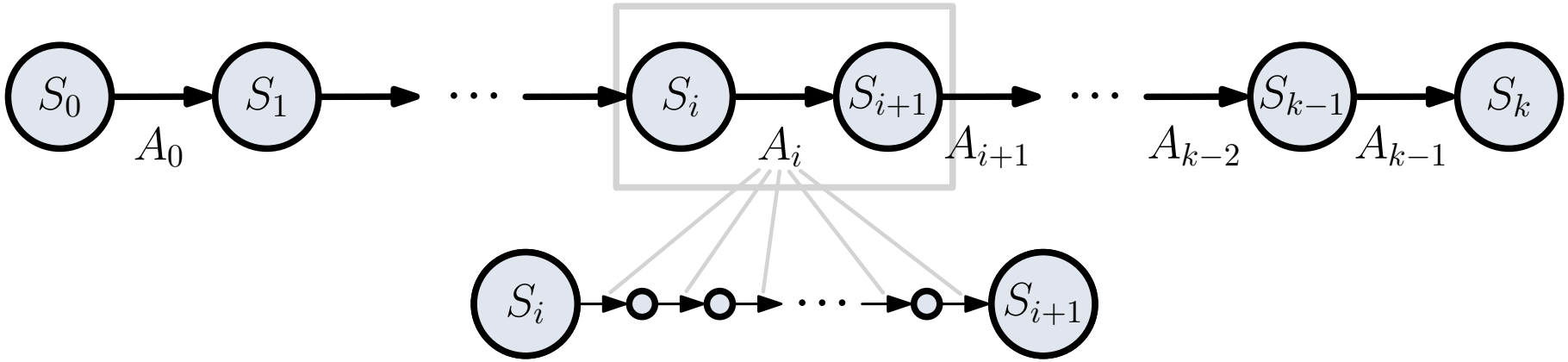
State variable is_p^t and action variable do_a^t for each step $t = 0 \ldots n$

1. Initial state $\mathcal{O}(|P|)$
2. Goal conditions $\mathcal{O}(|P|)$
3. Preconditions are satisfied at step t $\mathcal{O}(n \cdot |A| \cdot |E|)$
4. Effects hold at step $t + 1$ $\mathcal{O}(n \cdot |A| \cdot |E|)$
5. Frame axioms (state doesn't change arbitrarily) $\mathcal{O}(n \cdot |P|)$
6. Interference $\mathcal{O}(n \cdot |A|^2)$
   - Sequential action execution or $\forall$-step encoding
   - Abstraction: Nothing ($\exists$-step)

Fix Abstract Plan

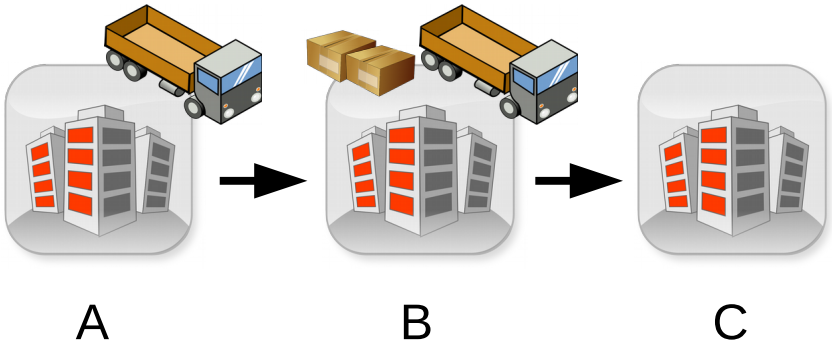


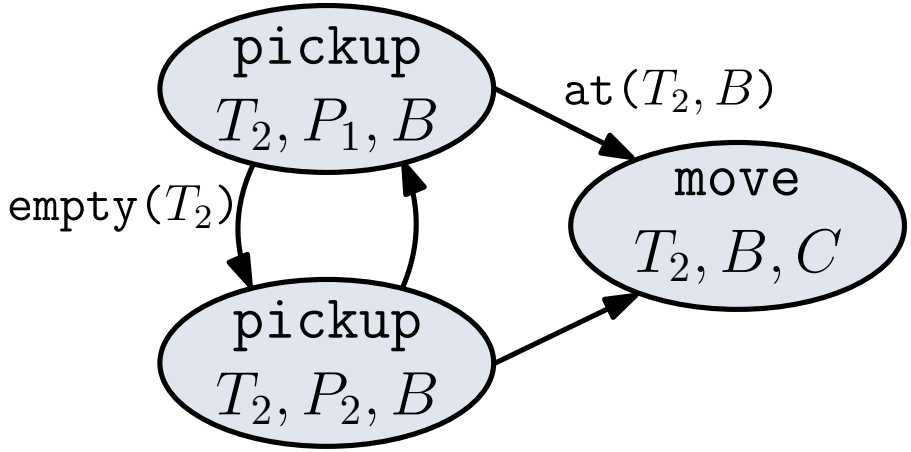
Fix Abstract Plan



1. Order actions
   - Topological ordering of disabling graph
2. Replan
   - GBFS from s_i to s_{i+1}
   - Small timeout
3. Refine
   - Action set is a counter example
   - Refine Abstraction

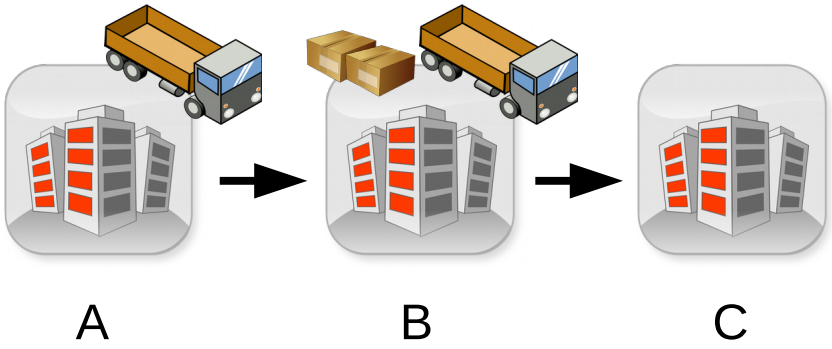
Trucking – Order

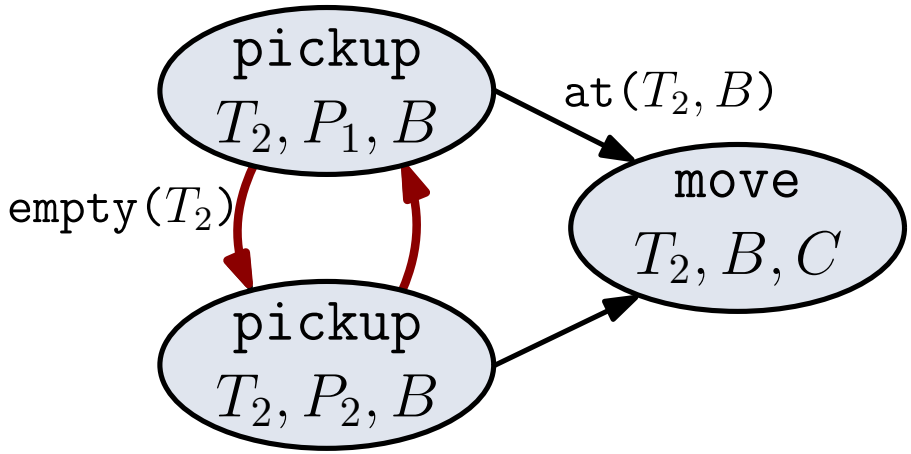




1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

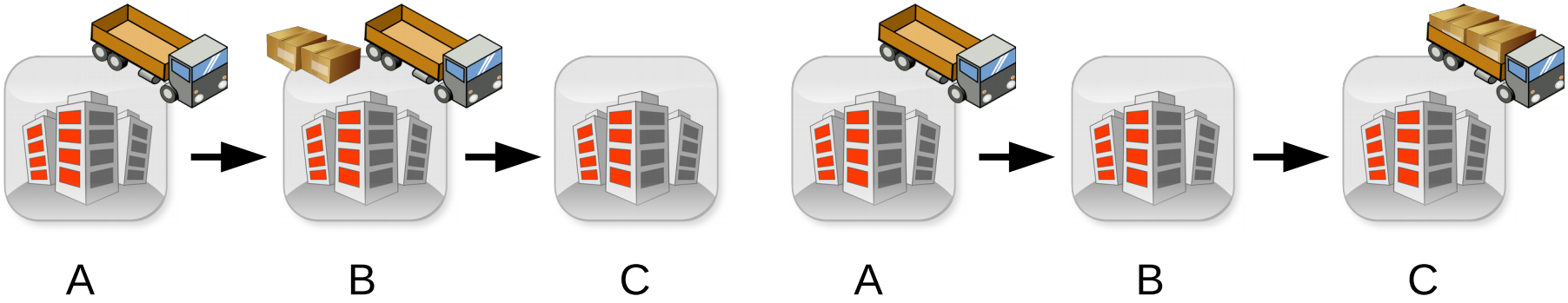
Trucking – Order





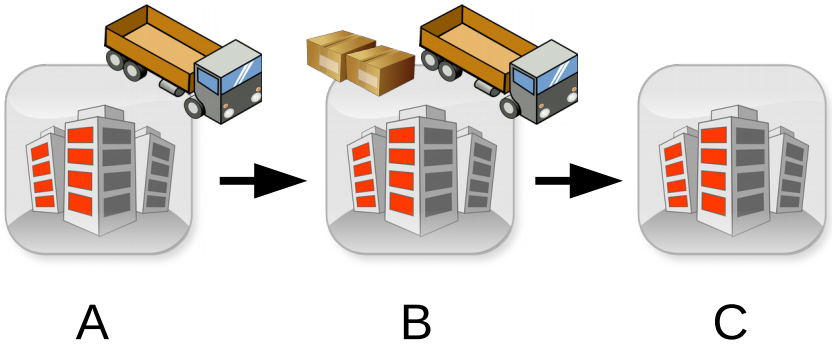
1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

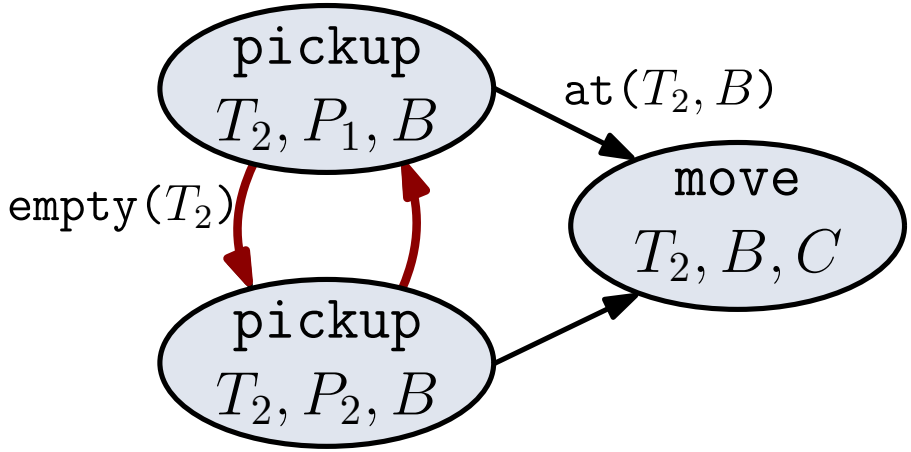
Trucking – Replan



1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

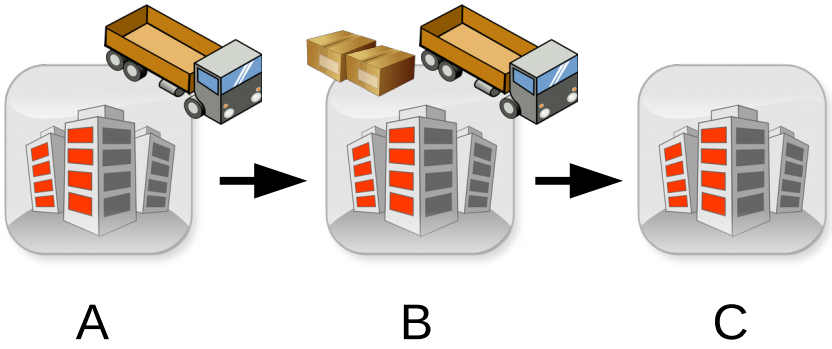
Trucking – Refine

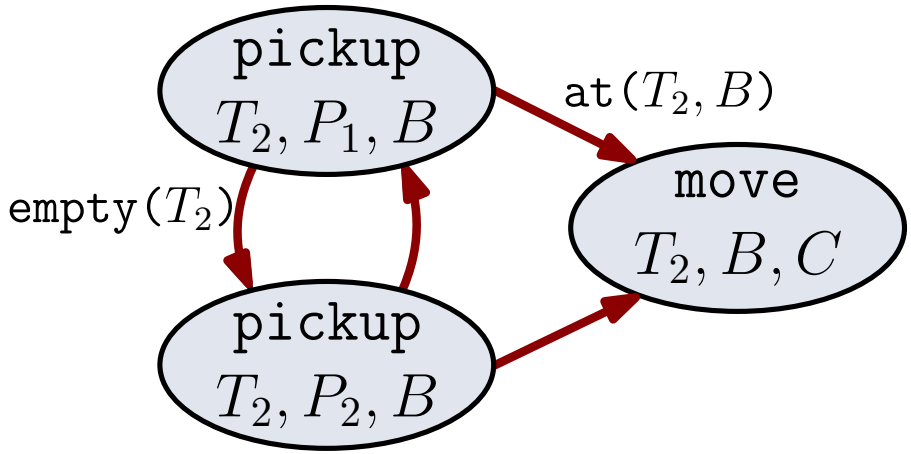




1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

Trucking – Refine

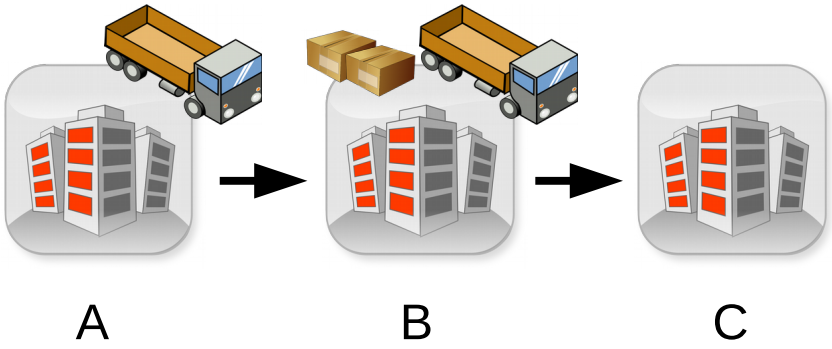


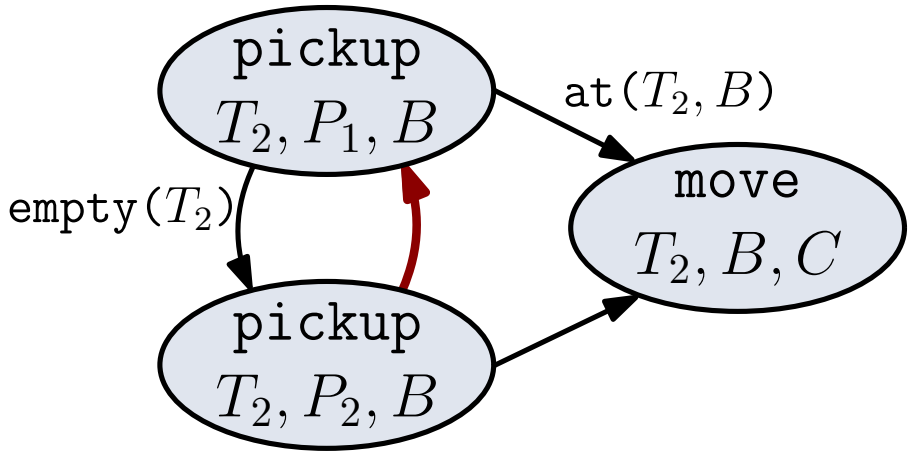


1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

- Add all edges $\rightarrow$ $\forall$-step semantics

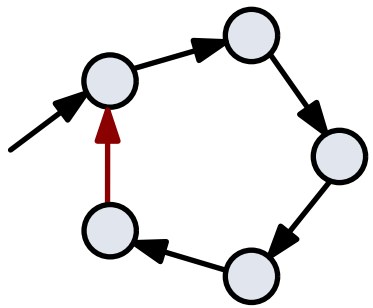
Trucking – Refine



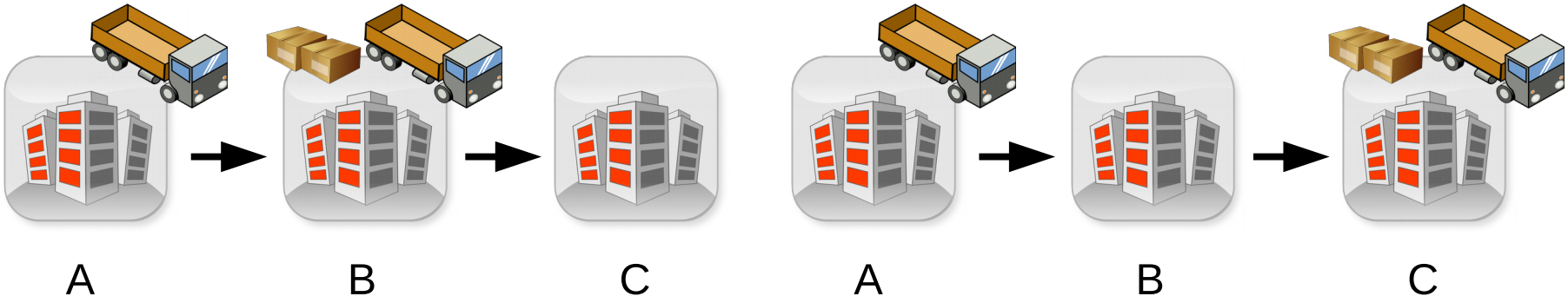


1. $\{\texttt{pickup}(T_2, P_1, B), \texttt{pickup}(T_2, P_2, B), \texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C), \texttt{drop}(T_2, P_2, C)\}$

- Add all edges $\rightarrow$ $\forall$-step semantics
- Mutex reduction: DFS back edges



Sparsification

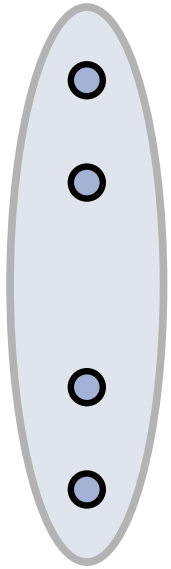


1. $\{\texttt{deliver}(T_2, P_1, B, C), \texttt{deliver}(T_2, P_2, B, C)\}$

Planning problem is overdefined

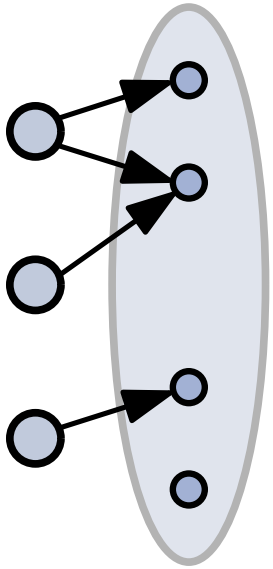
$\Rightarrow$ Plan sparsification

Sparsification



- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

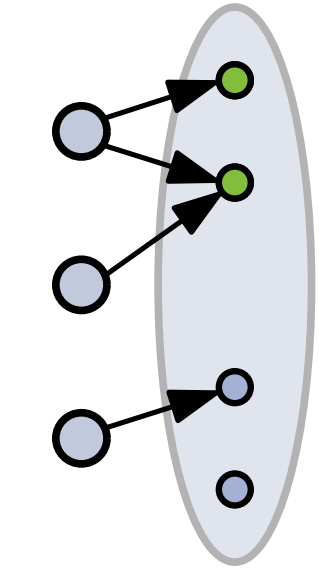
Sparsification



A_{k-1}

- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

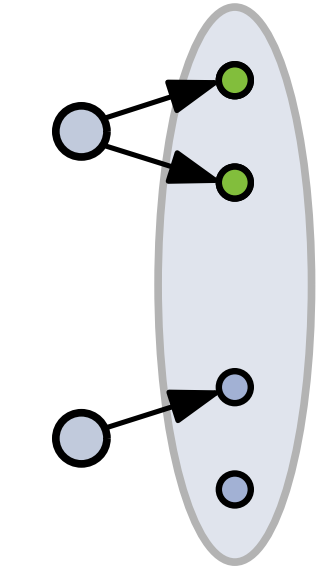
Sparsification



A_{k-1}

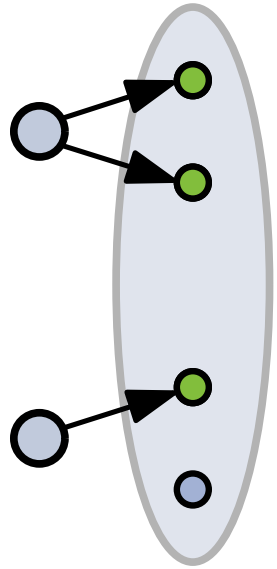
- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

Sparsification



- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

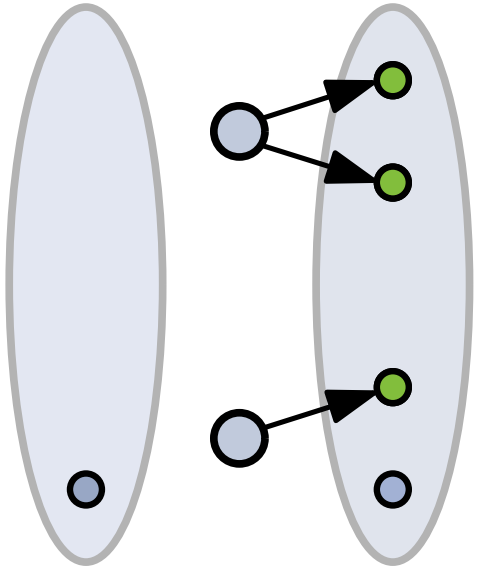
Sparsification



A_{k-1}

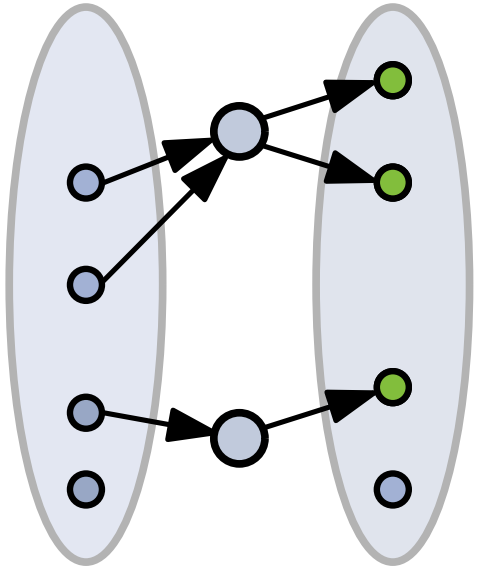
- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

Sparsification



- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

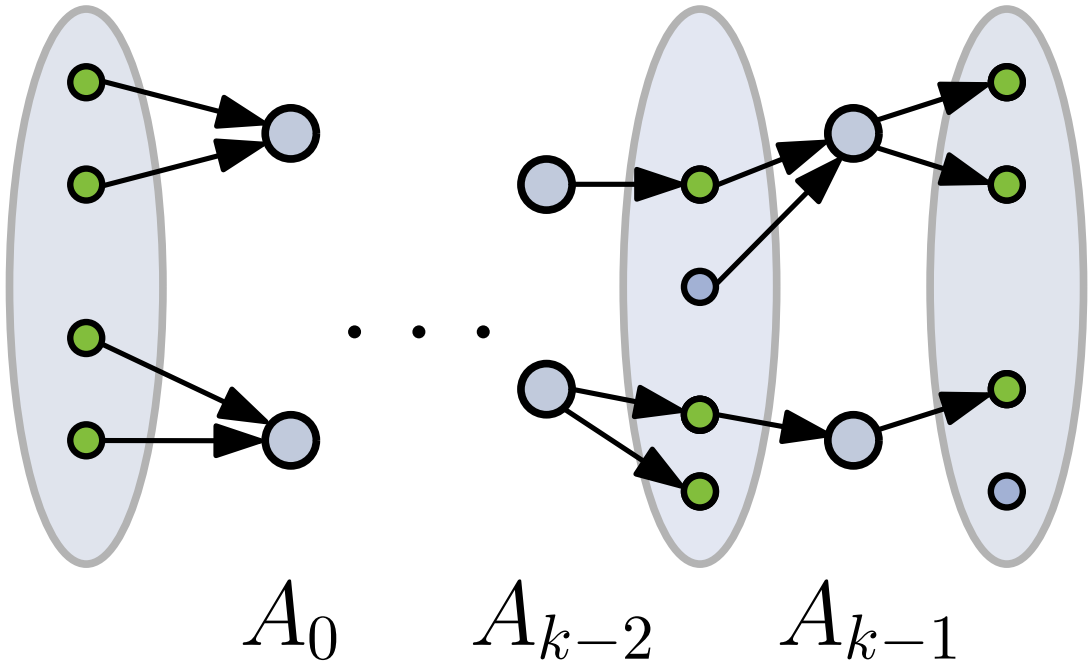
Sparsification



A_{k-1}

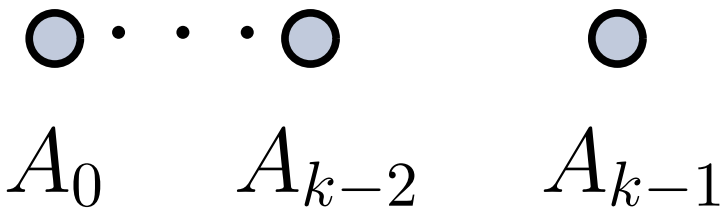
- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

Sparsification



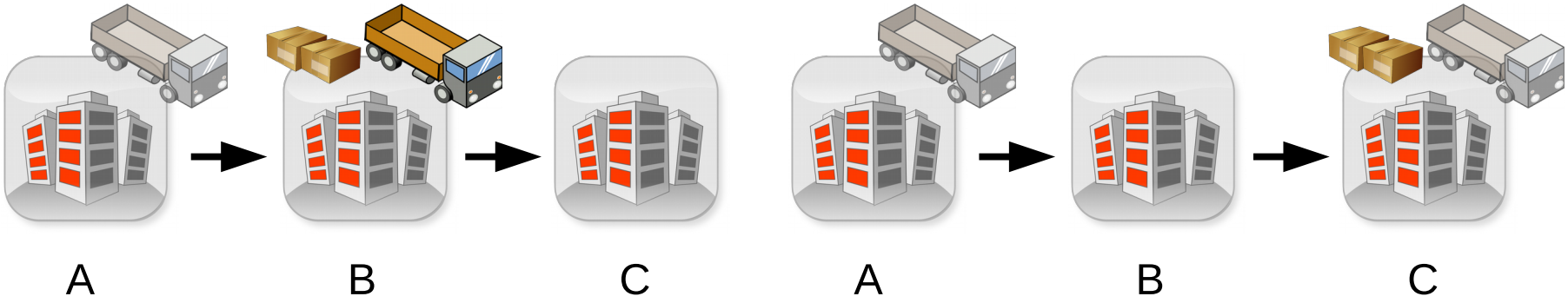
- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

Sparsification



- Plan sparsification
- Greedy Action elimination algorithm [Nakhost and Müller, 2010]

Sparsification



1. $\{\texttt{deliver}(T_2, P_1, B, C), \texttt{deliver}(T_2, P_2, B, C)\}$

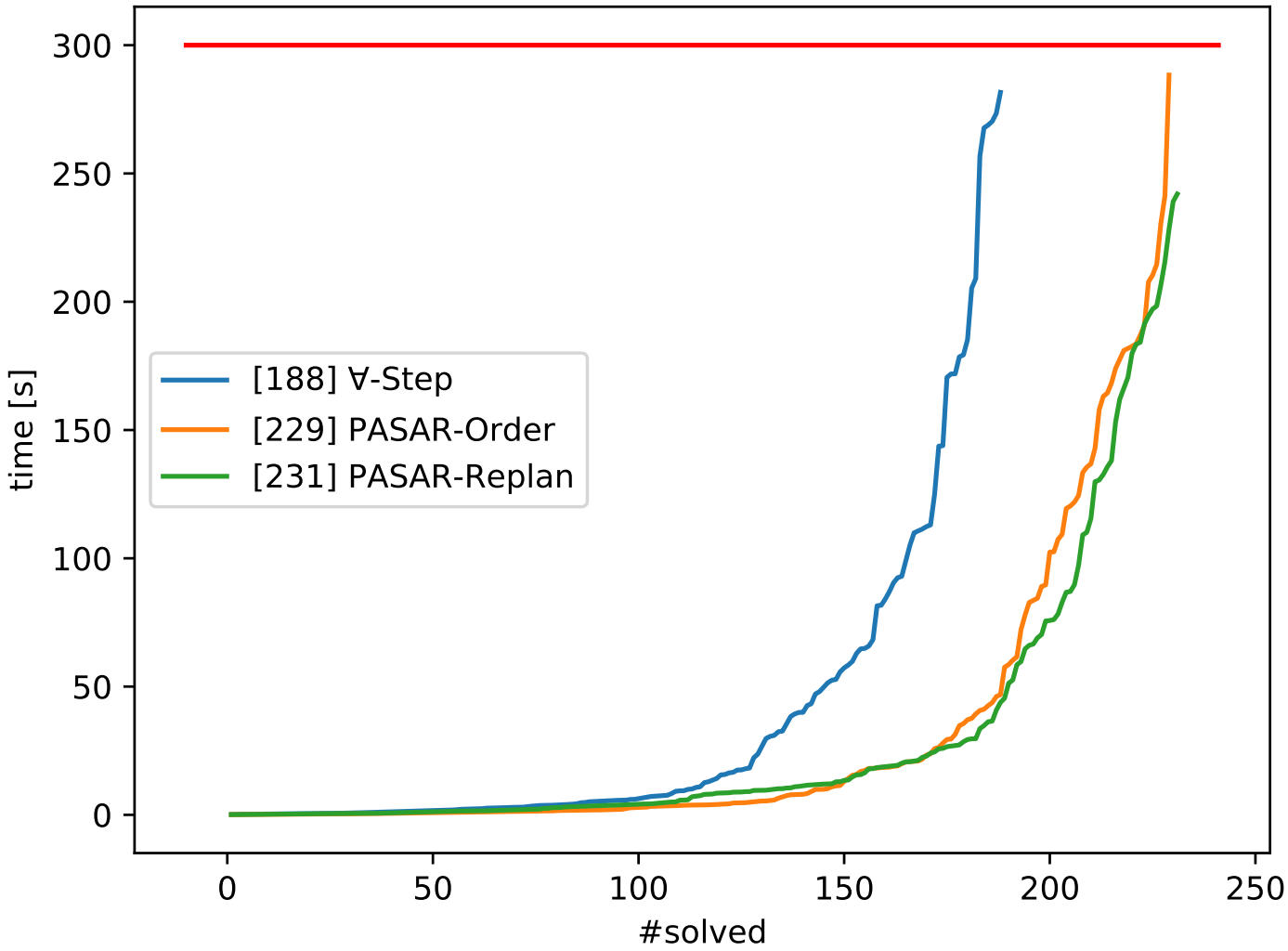
Implementation Details

- Fast Downward to ground to *SAS+* [Helmert, 2006]
- *IPASIR* generic interface for incremental SAT solving [Balyo et al., 2016]
- We chose Glucose [Audemard and Simon, 2009]

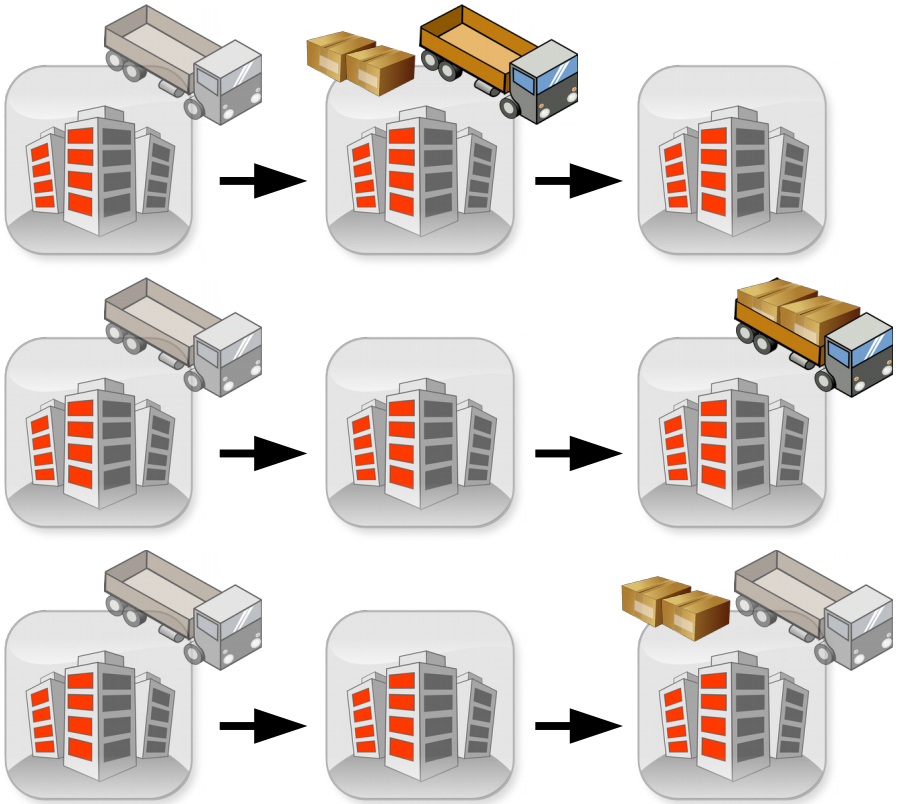
Evaluation I

- Benchmarks from the satisficing and optimal tracks of *IPC* 2014 and 2018
- Up to 5 minutes of total run time
- State space search limited to 0.2 seconds
- Competitors:
  - $\forall$-Step encoding using our translation
  - PASAR-Order only ordering
  - PASAR-Replan ordering and replan

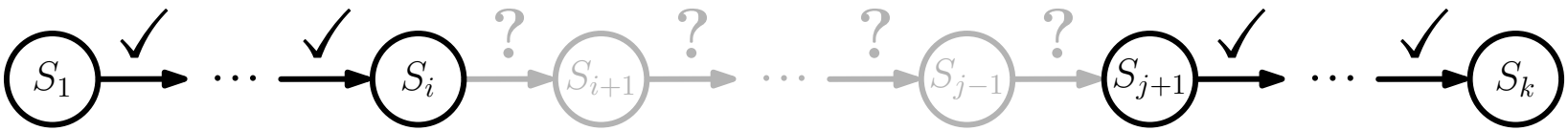
Evaluation I



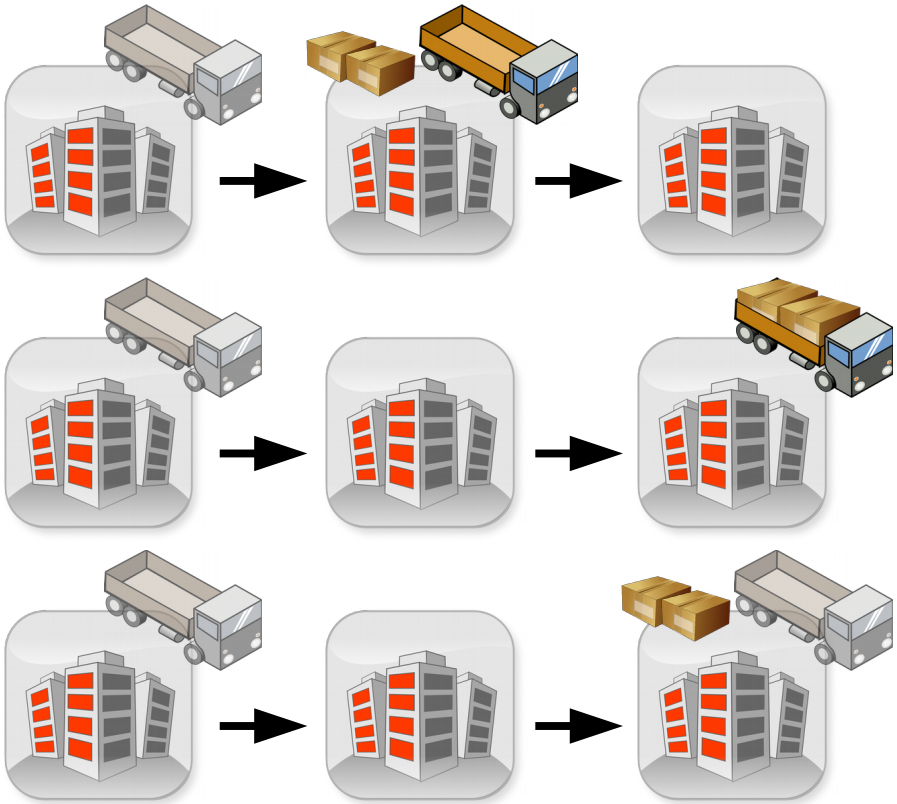
Step Skipping



1. $\{\texttt{pickup}(T_2, P_1, B),$
   $\texttt{pickup}(T_2, P_2, B),$
   $\texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C),$
   $\texttt{drop}(T_2, P_2, C)\}$

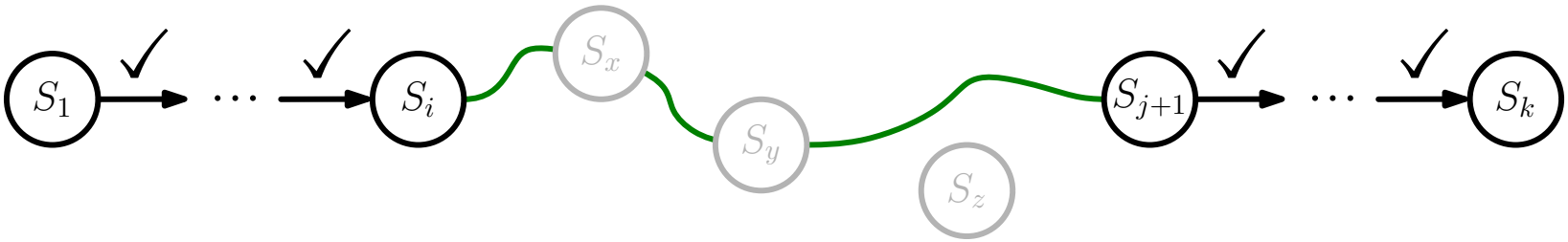


Step Skipping – local

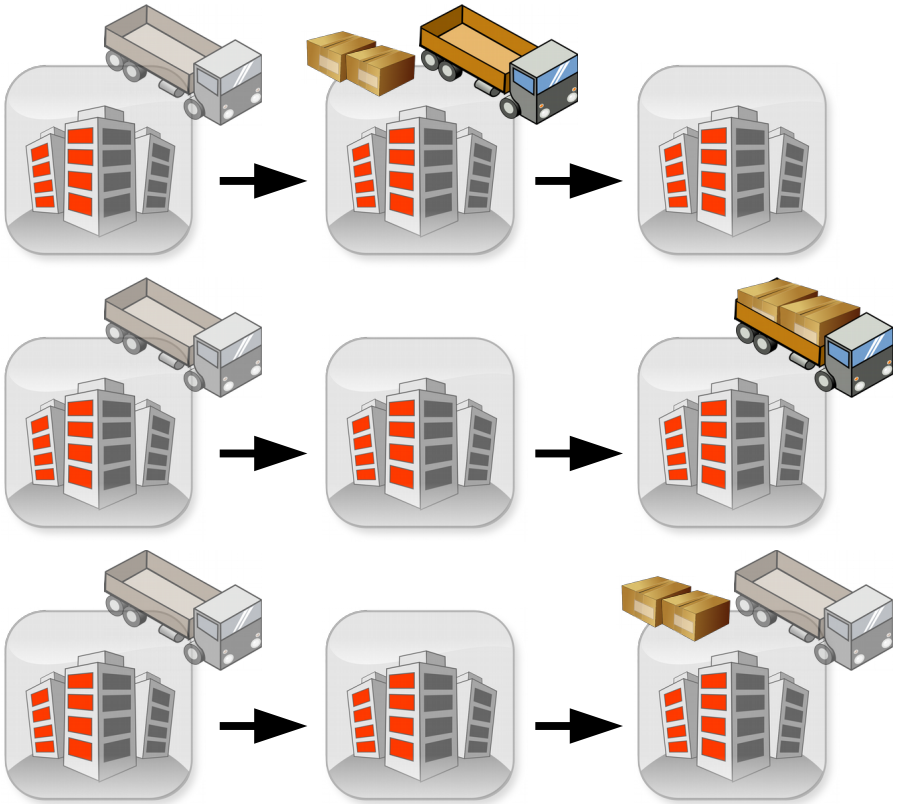


1. $\{\texttt{pickup}(T_2, P_1, B),$
   $\texttt{pickup}(T_2, P_2, B),$
   $\texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C),$
   $\texttt{drop}(T_2, P_2, C)\}$

$$h(s) = \sum_{x=i+1}^{j+1} d_M(s, s_x) \cdot 1.2^x$$

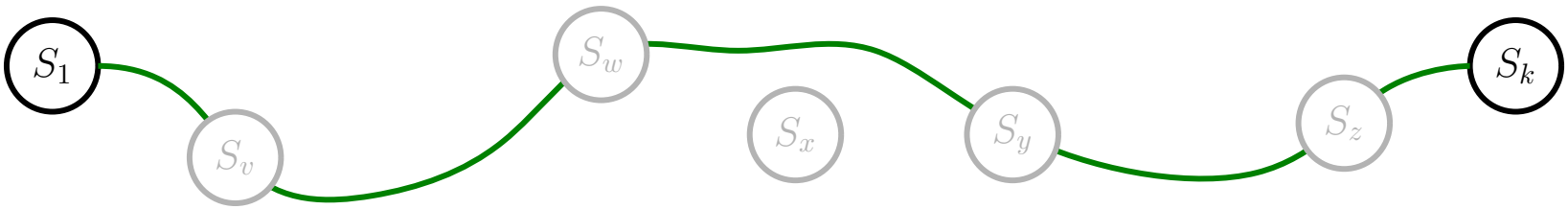


Step Skipping – global



1. $\{\texttt{pickup}(T_2, P_1, B),$
   $\texttt{pickup}(T_2, P_2, B),$
   $\texttt{move}(T_2, B, C)\}$
2. $\{\texttt{drop}(T_2, P_1, C),$
   $\texttt{drop}(T_2, P_2, C)\}$

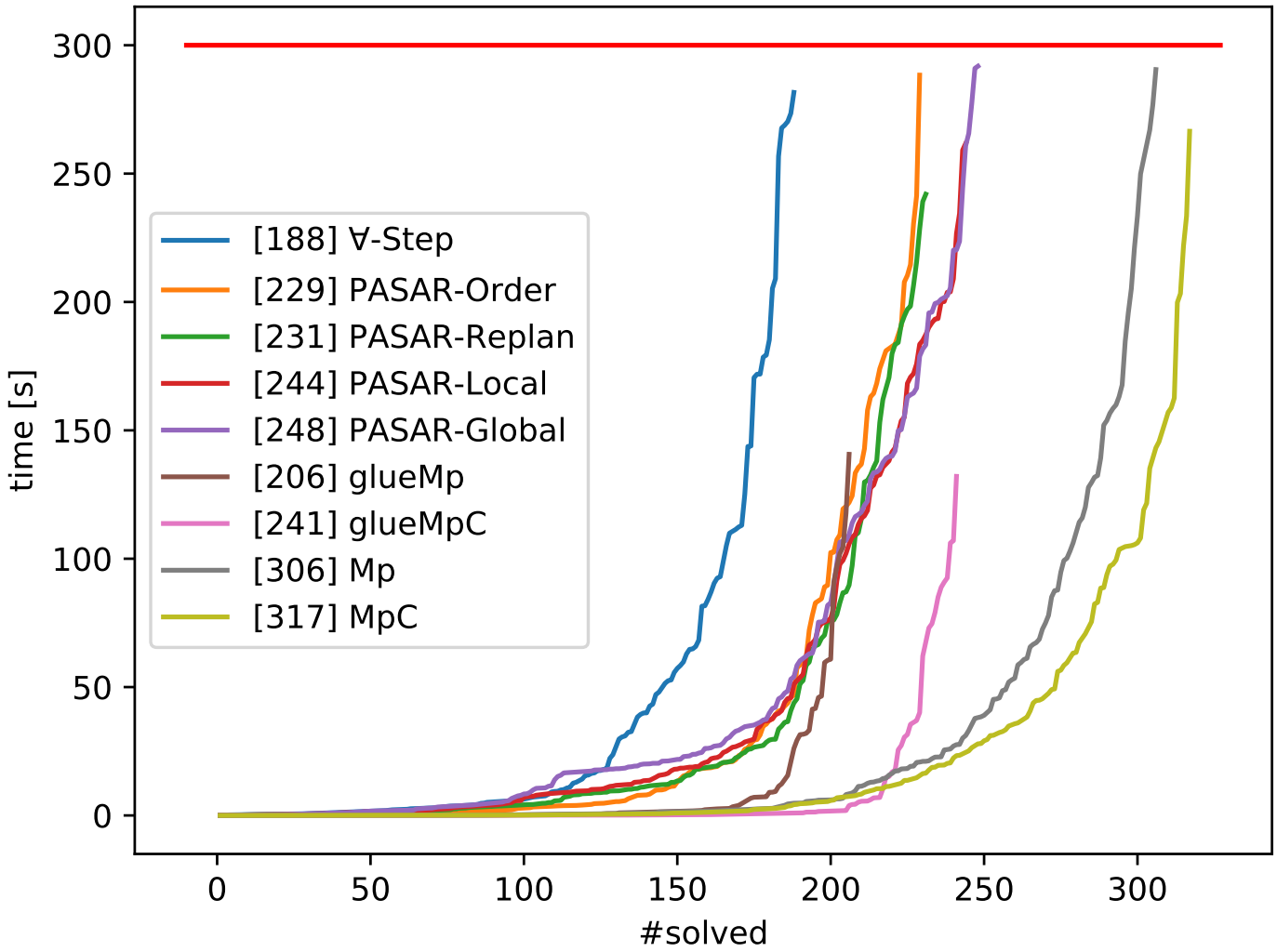
$$h(s) = \sum_{x=2}^{k} \quad d_M(s, s_x) \cdot 1.2^x$$



Evaluation II

- Competitors:
  - ∀-Step encoding using our translation
  - PASAR-Order only ordering
  - PASAR-Replan ordering and replan
  - PASAR-Local local step skipping
  - PASAR-Global additionally global step skipping fallback
  - Madagascar [Rintanen, 2013]
    - glueMp Madagascar (linear scheduling) with glucose
    - glueMpC Madagascar (exponential scheduling) with glucose
    - Mp Madagascar (linear scheduling)
    - MpC Madagascar (exponential scheduling)

Evaluation II



Conclusion

Sparkle Planning Challenge (ICAPS 19)

- Contributed 22% to the optimal portfolio
- 3rd most relevant planner

Future Work

- More coarse-grained Abstraction with increasing Refinement
- Relaxed encodings
- Action learning

References I

Audemard, G. and Simon, L. (2009).

Predicting learnt clauses quality in modern SAT solvers.

In *Twenty-first International Joint Conference on Artificial Intelligence*.

Balyo, T., Biere, A., Iser, M., and Sinz, C. (2016).

SAT Race 2015.

Artificial Intelligence, 241:45–65.

Helmert, M. (2006).

The fast downward planning system.

Journal of Artificial Intelligence Research (JAIR), 26:191–246.

Kautz, H. and Selman, B. (1992).

Planning as Satisfiability.

In *Proceedings of the European Conference on Artificial Intelligence*, pages 359–363.

References II

Nakhost, H. and Müller, M. (2010).

Action elimination and plan neighborhood graph search: Two algorithms for plan improvement.

pages 121–128.

Rintanen, J. (2013).

Planning as satisfiability: state of the art.

`https://users.aalto.fi/rintanj1/satplan.html`.

Rintanen, J., Heljanko, K., and Niemelä, I. (2006).

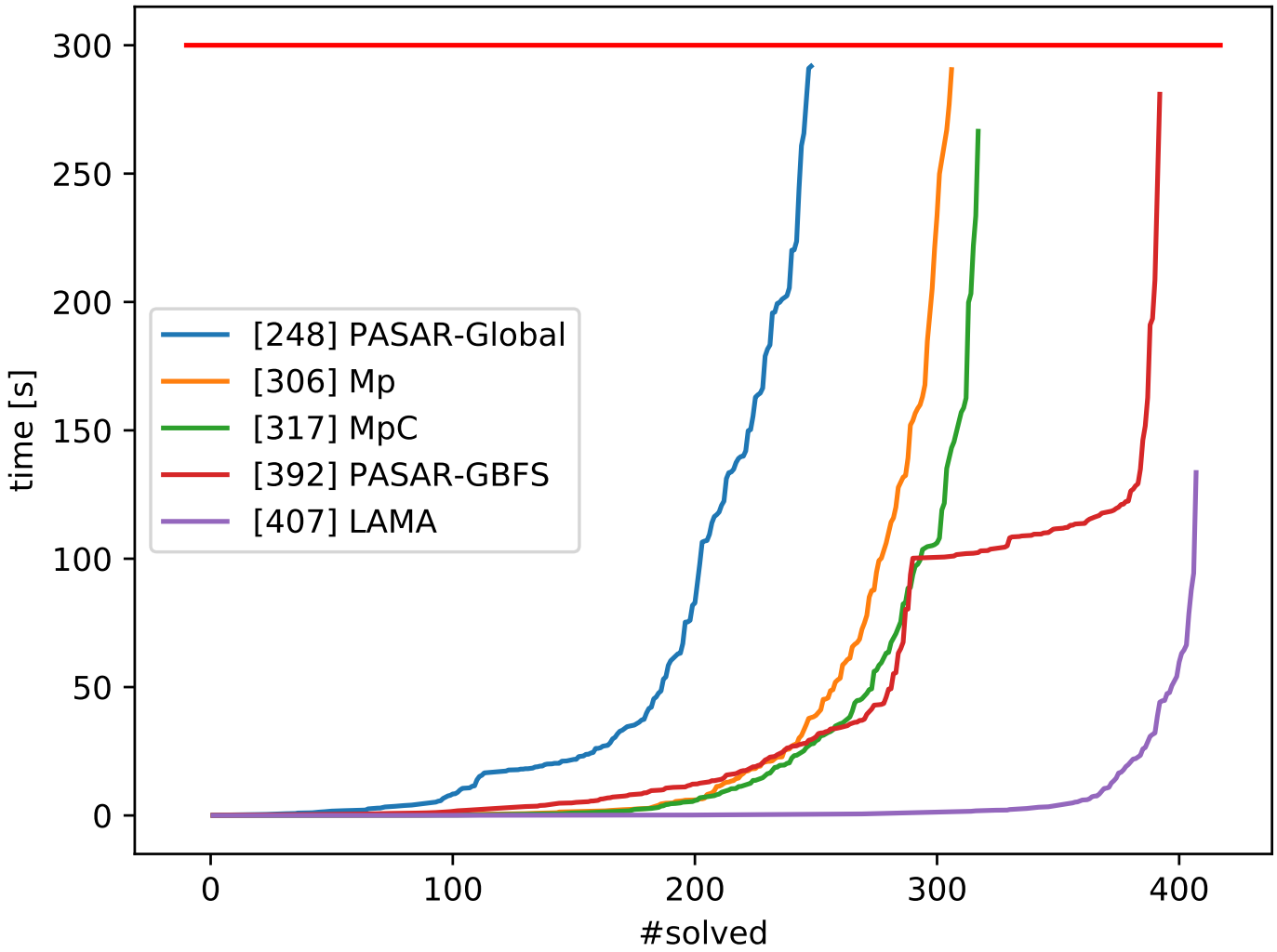
Planning as satisfiability: parallel plans and algorithms for plan search.

Artificial Intelligence, 170(12-13):1031–1080.

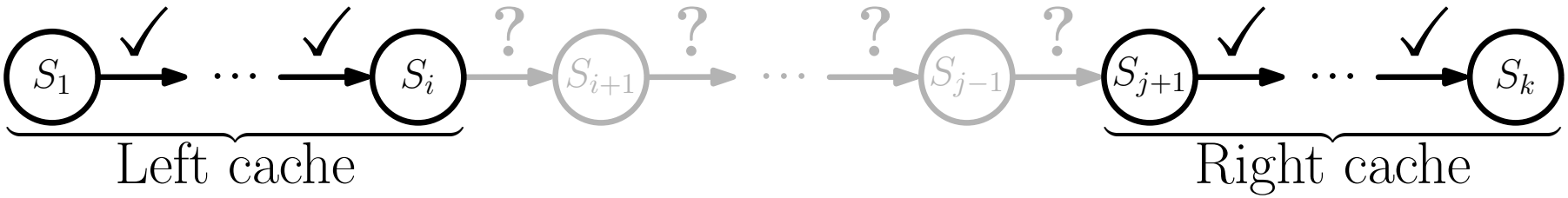
Evaluation III

- Competitors:
  - ∀-Step encoding using our translation
  - PASAR-Order only ordering
  - PASAR-Replan ordering and replan
  - PASAR-Local local step skipping
  - PASAR-Global additionally global step skipping fallback
  - Madagascar [Rintanen, 2013]
    - glueMp Madagascar (linear scheduling) with glucose
    - glueMpC Madagascar (exponential scheduling) with glucose
    - Mp Madagascar (linear scheduling)
    - MpC Madagascar (exponential scheduling)
  - PASAR-GBFS initial GBFS for 100 seconds
  - LAMA Fast Downward with the LAMA configuration [Helmert, 2006]

Evaluation III



Plan Caching



- Plan from initial state is known
- Order and replan
- Reached by state skipping search

- Plan to goal state is known
- Only order (no full state known)

Same Makespan Limit

- The abstraction doesn't represent enough of the problem
- When too many abstractions are solved in the same makespan
- Add all remaining interferences

  $\Rightarrow$ Fallback to $\forall$-step