Planning with Unknown Object Quantities and Properties

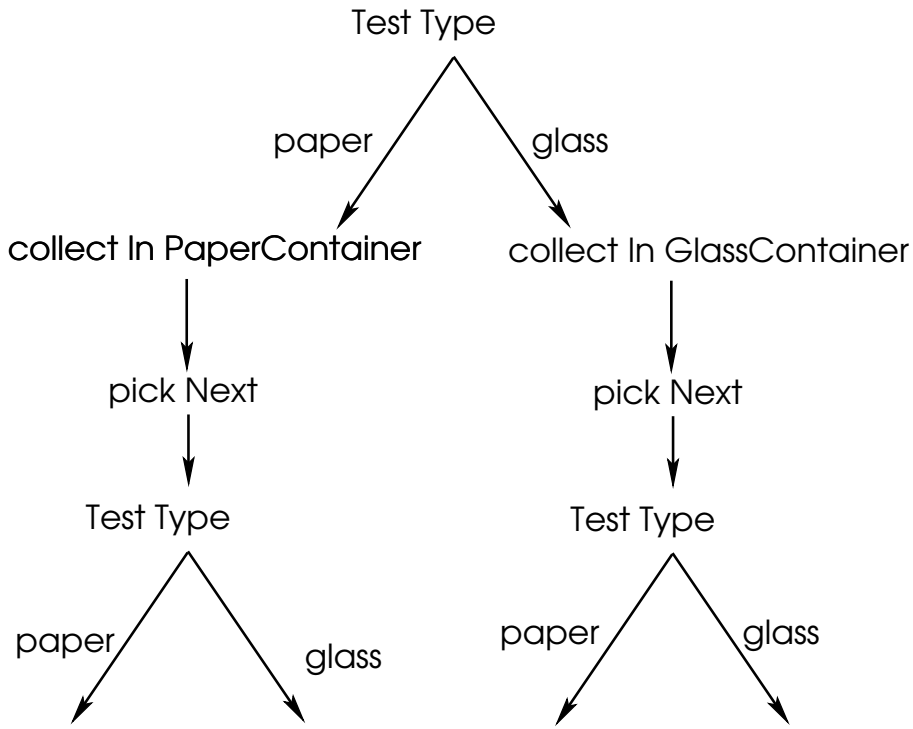
Siddharth Srivastava
Joint work with Neil Immerman and Shlomo Zilberstein
University of Massachusetts, Amherst

Eighth Symposium on Abstraction, Reformulation and Approximation
7 – 10 July, 2009

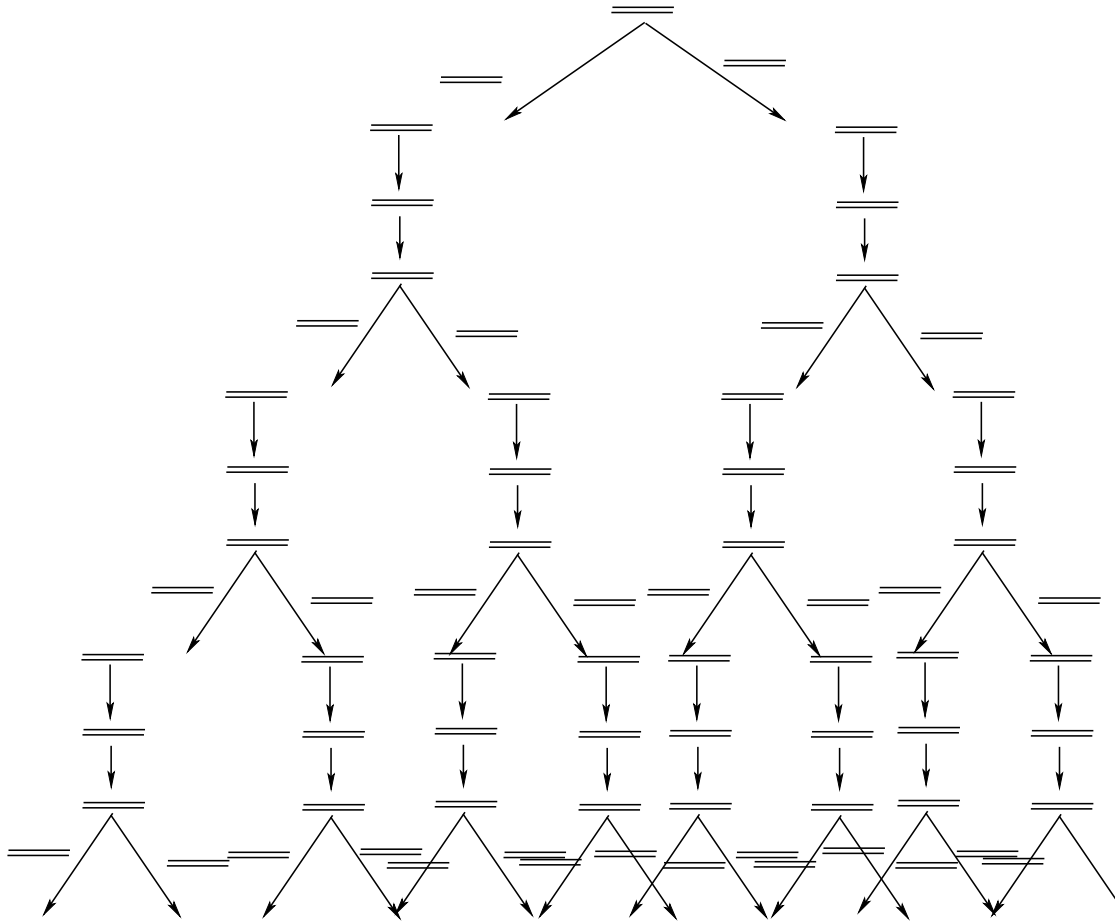
Overview

- Introduction
- Framework
  - Concrete Representation
  - Abstract Representation for Belief States
  - Actions on Belief States
- Planning Algorithms
  - Plan Generalization
  - Plan Merging
  - Preconditions
- Results

Conditional Planning

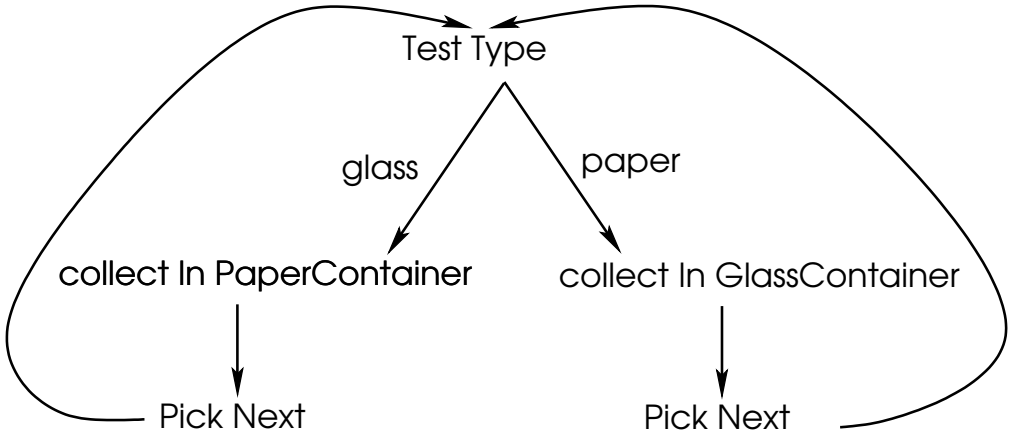


Conditional Planning



Need to merge, maintain history

Conditional Planning

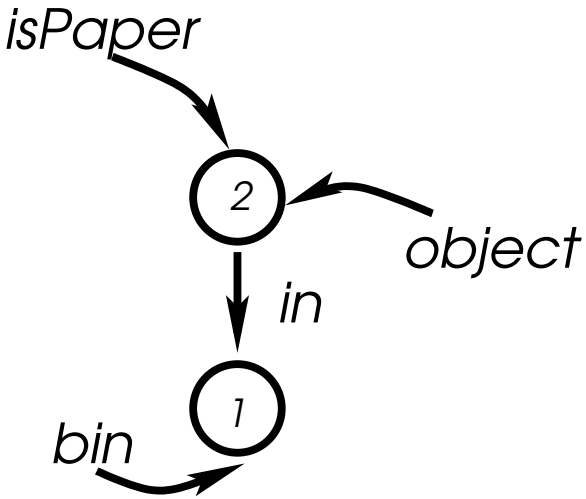


Need to merge, maintain history

Progress and termination of loops

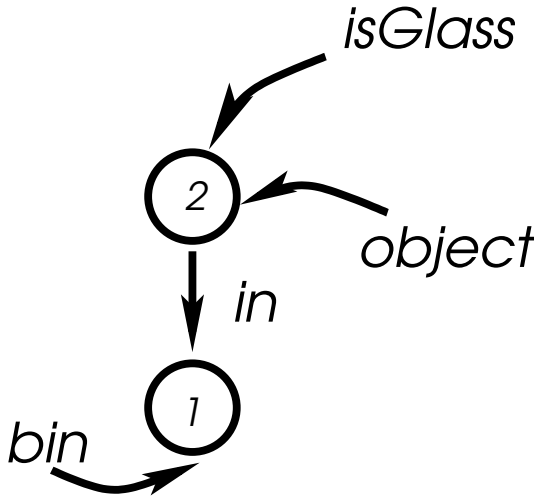
Concrete States as First-Order Structures

$$\mathcal{V} = \{object^1, bin^1, isGlass^1, isPaper^1, in^2, empty^1, collected^1, forGlass^1, forPaper^1\}$$



((object(2))) =1
((isPaper(2))) = 1
((bin(1))) = 1
((in(2,1))) = 1

s_1



((object(2))) =1
((isGlass(2))) = 1
((bin(1))) = 1
((in(2,1))) = 1

s_2

Action Operators

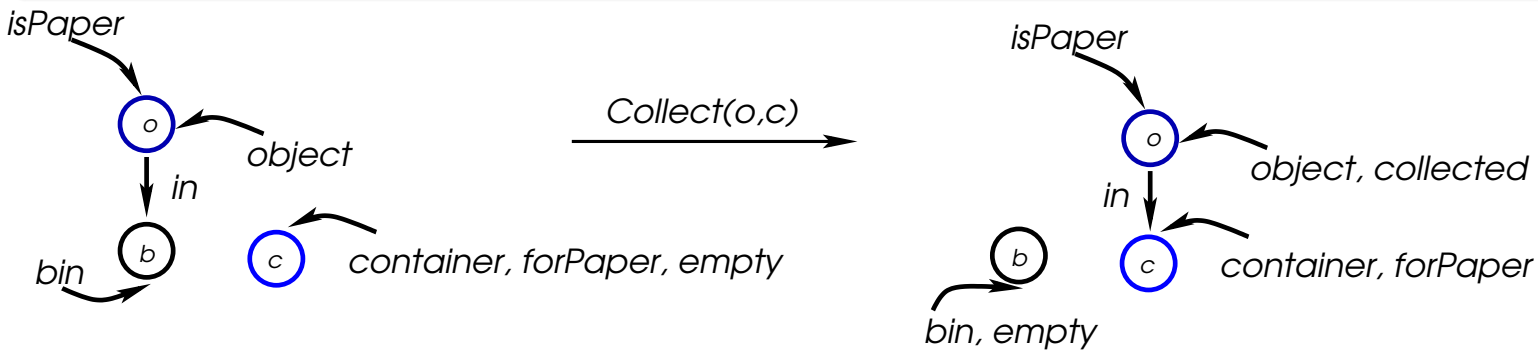
- Precondition: formula in FO.
- Predicate updates
  - $p'(\bar{x}) = \overbrace{(\neg p(\bar{x}) \wedge \Delta_p^+(\bar{x}))}^{\text{tuples added to } p} \vee \overbrace{(p(\bar{x}) \wedge \neg \Delta_p^-(\bar{x}))}^{\text{tuples retained in } p}$
- Use formula evaluation to compute action effect.
- Frame axioms/successor state axioms (situation calculus) using a double vocabulary.

Example: The Collect Action

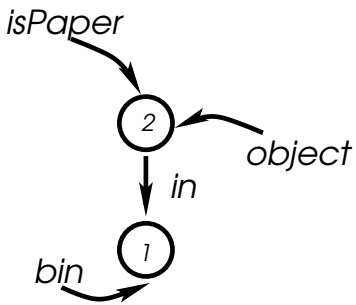
Collect(o,c)

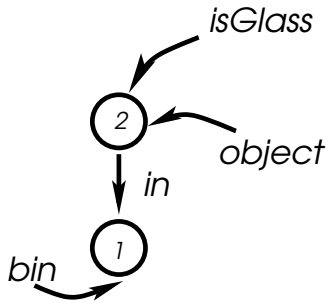
$\text{object}(o) \wedge \text{container}(c) \wedge (\text{isGlass}(o) \leftrightarrow \text{forGlass}(c)) \wedge$
$\exists b(\text{bin}(b) \wedge \text{in}(o,b) \wedge \text{robotAt}(b))$

$$
\begin{aligned}
in'(u,v) &:= (in(u,v) \wedge u \neq o) \vee \\
&\quad (\neg in(u,v) \wedge u = o \wedge v = c) \\
empty'(u) &:= (empty(u) \wedge u \neq c) \vee in(o,u) \\
collected'(u) &:= collected(u) \vee o = u
\end{aligned}
$$

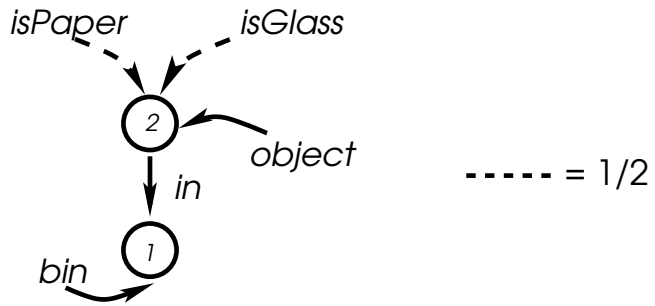


Abstraction Using 3-Valued Logic



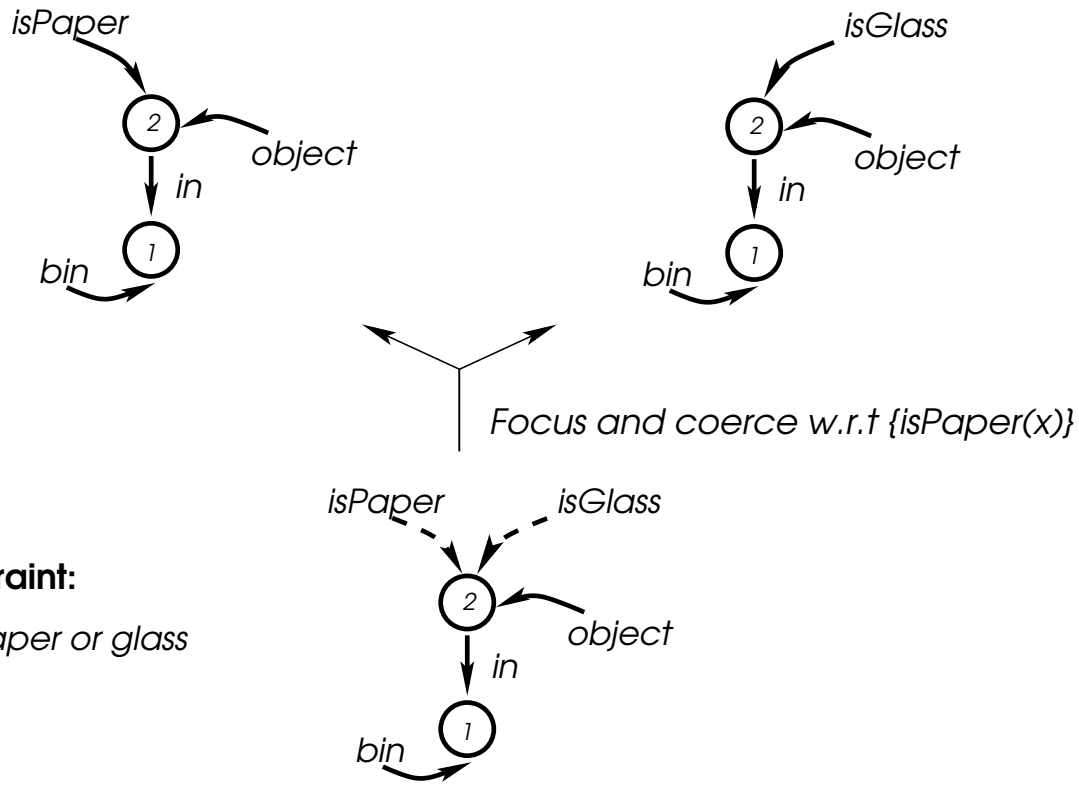


Use 3–Valued logic to abstract as:



TVLA: [Sagiv et al., 2002]

Abstraction Using 3-Valued Logic

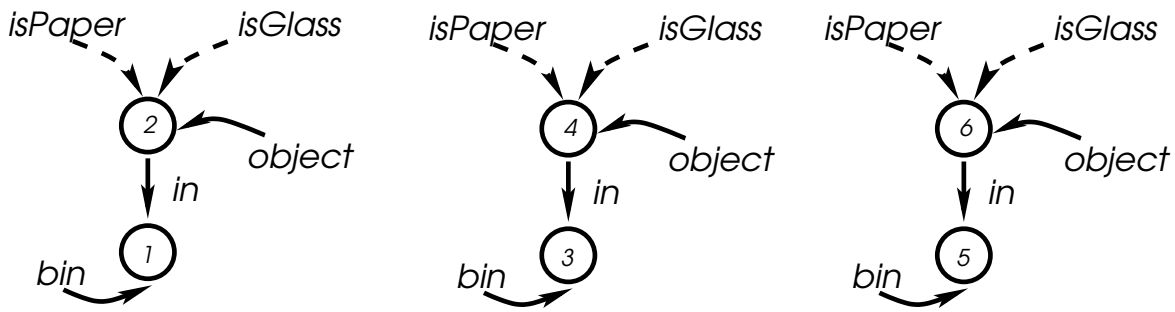


Integrity Constraint:

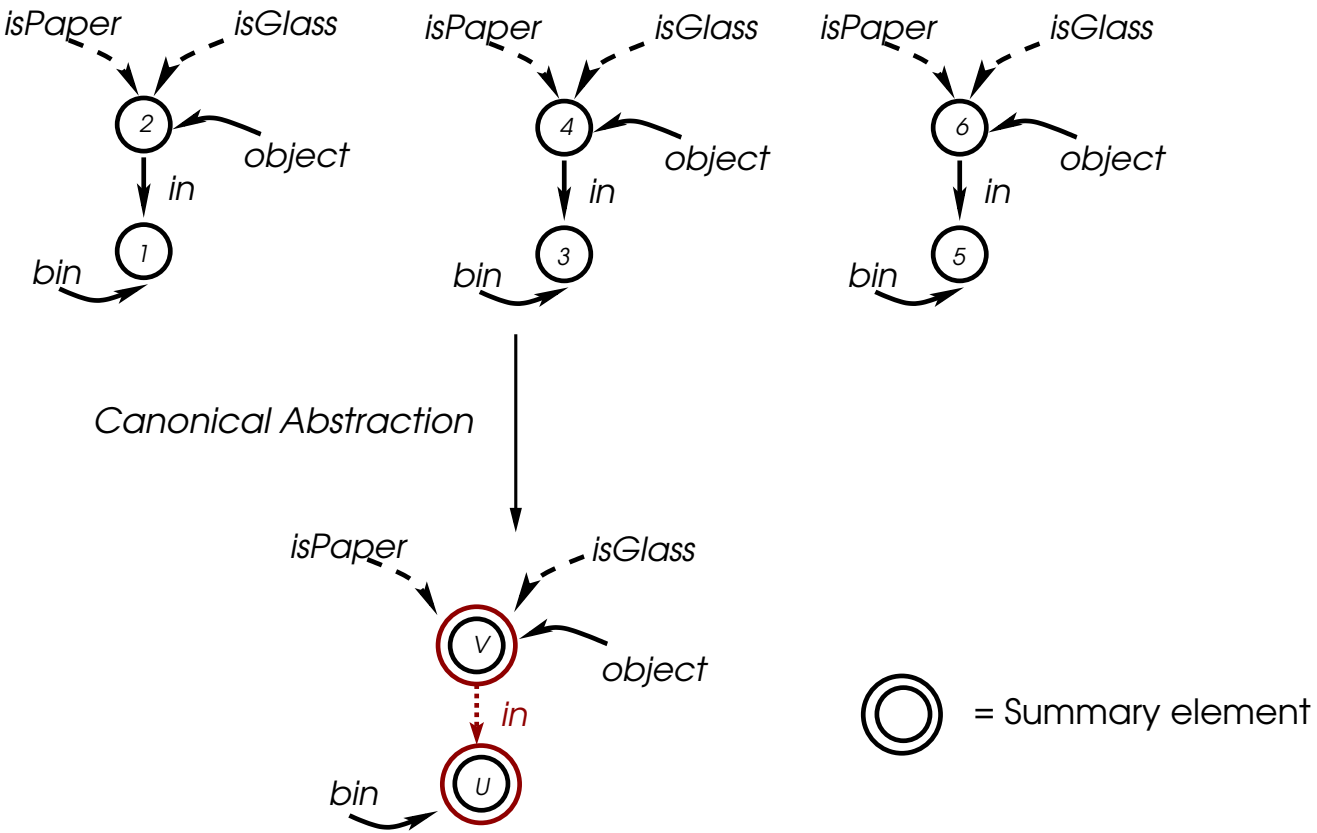
Objects are either paper or glass

Implementation of "sensing" actions

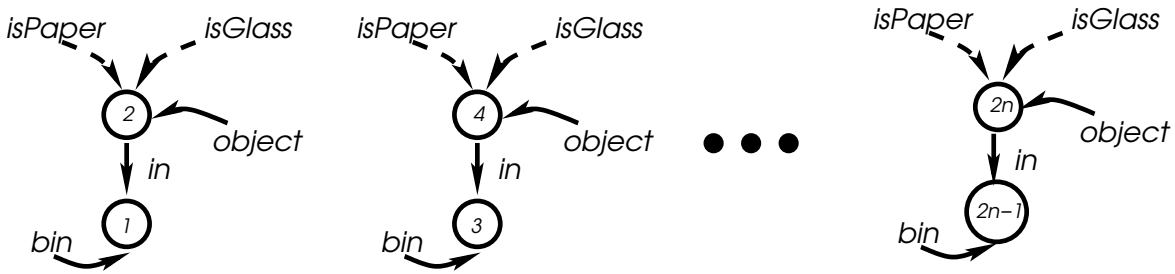
Abstraction Using 3-Valued Logic

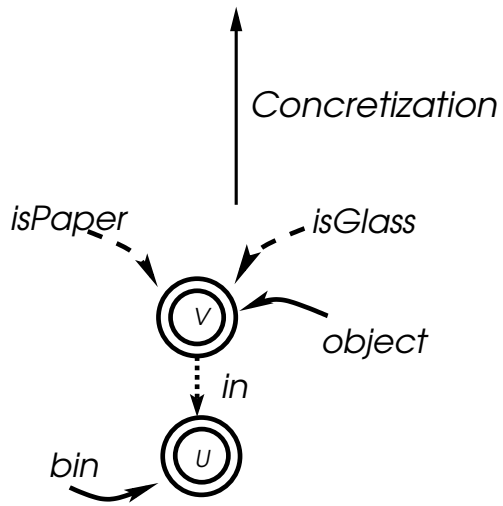


Abstraction Using 3-Valued Logic



Abstraction Using 3-Valued Logic





Integrity Constraint:

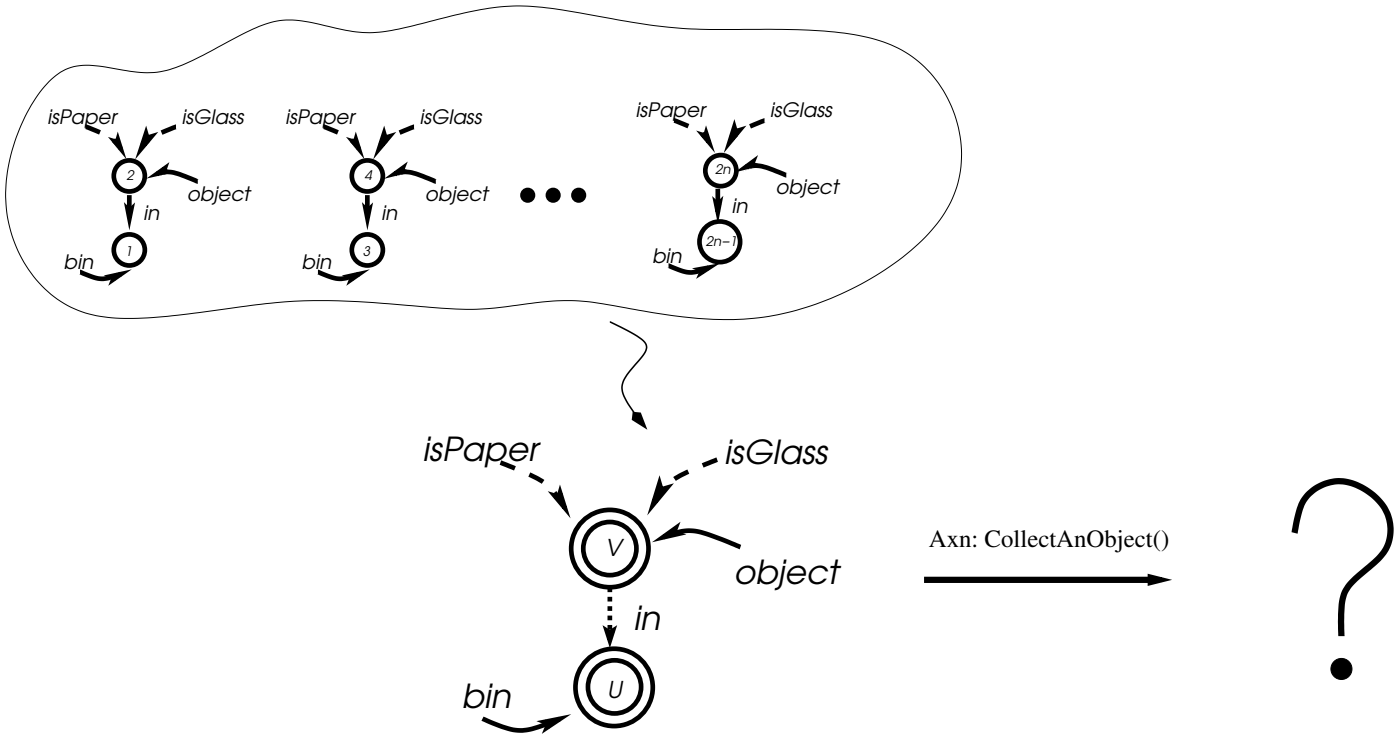
Each bin has a unique object

Abstraction Using 3-Valued Logic: Summary

TVLA [Sagiv et al., 2002]: Three Valued Logic Analysis

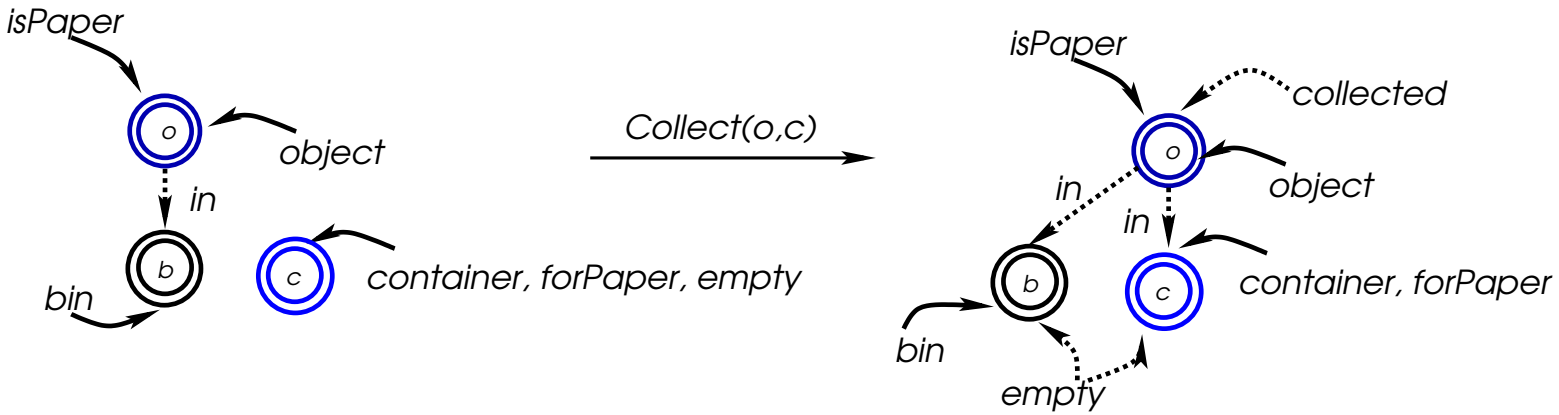
- Abstraction predicates: unary predicates.
- Element's role = set of abstraction predicates satisfied
- Collapse elements of a role into summary elements.
- Use integrity constraints to retreive concrete states.

Action Application on Belief States



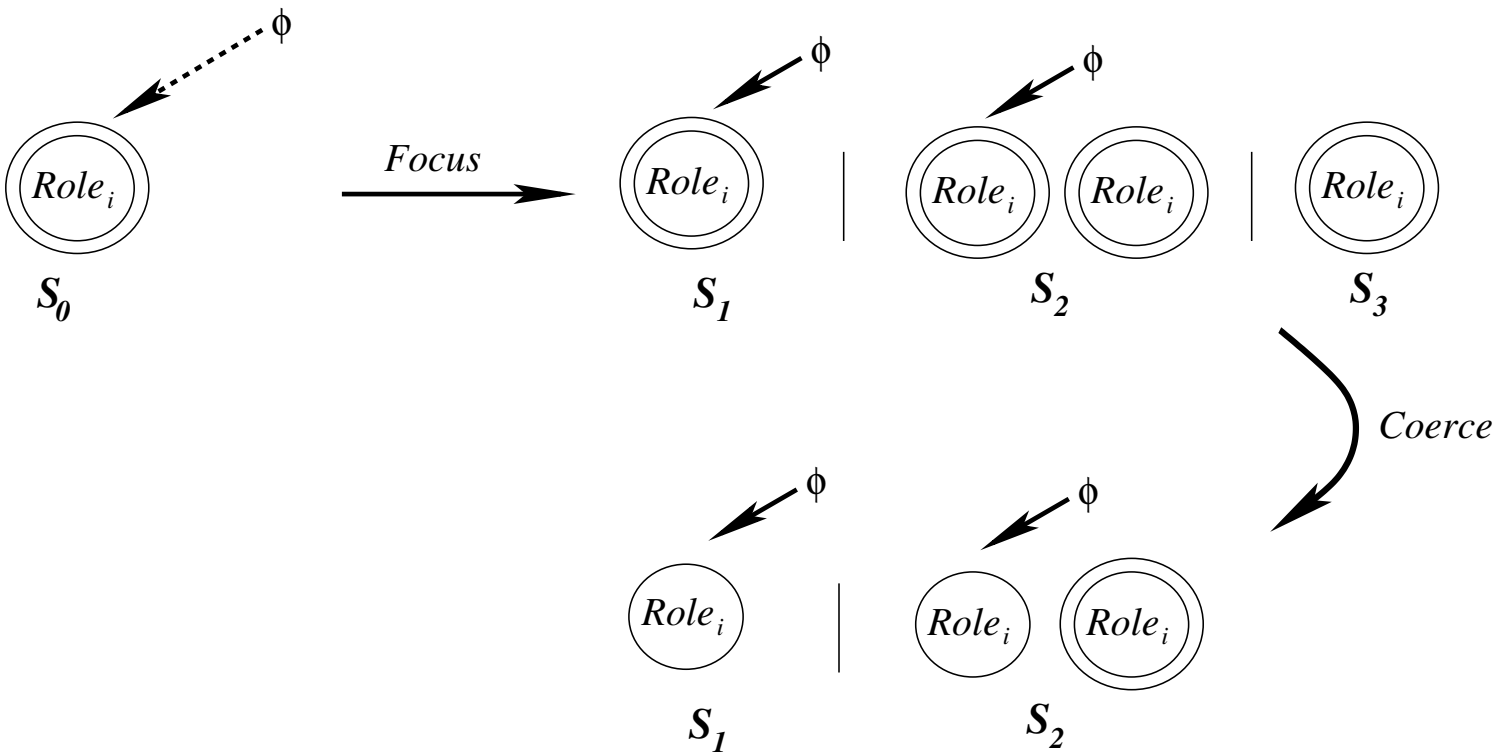
Action Application on Belief States

A simpler case: only paper objects and containers



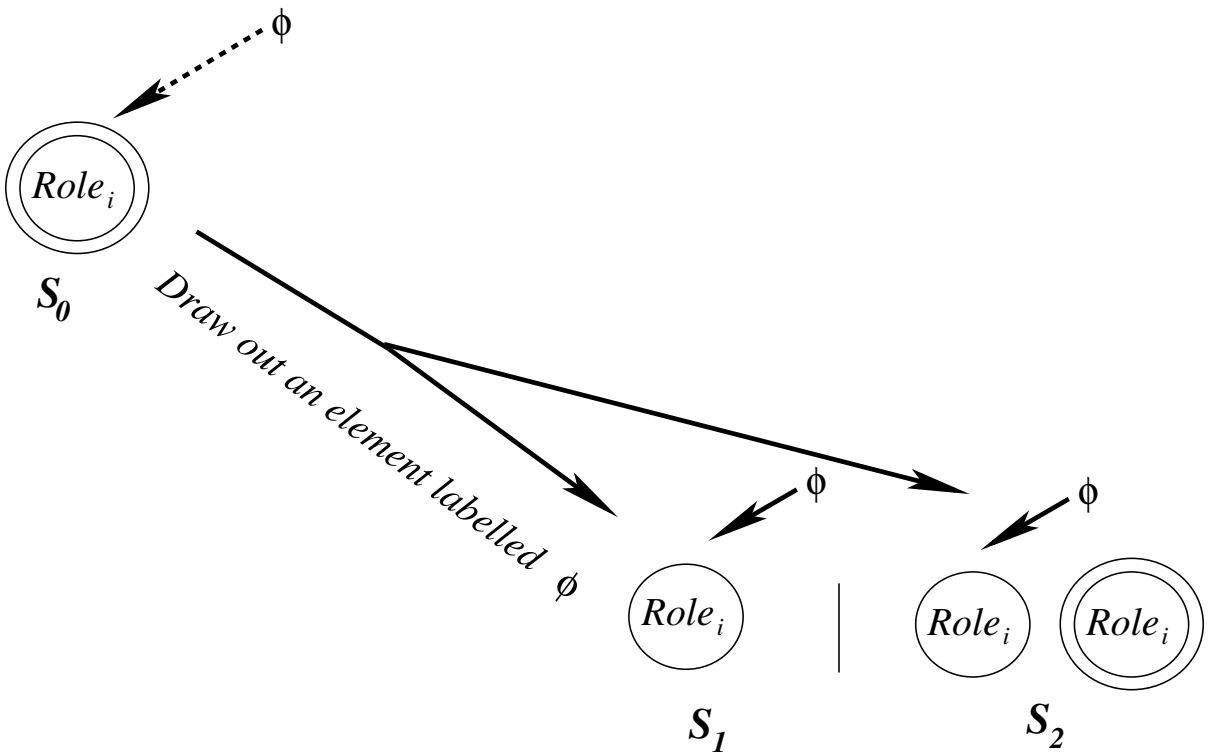
Underspecified action application

Drawing Out Action Arguments



ϕ constrained to be unique and satisfiable

Drawing Out Action Arguments



ϕ constrained to be unique and satisfiable

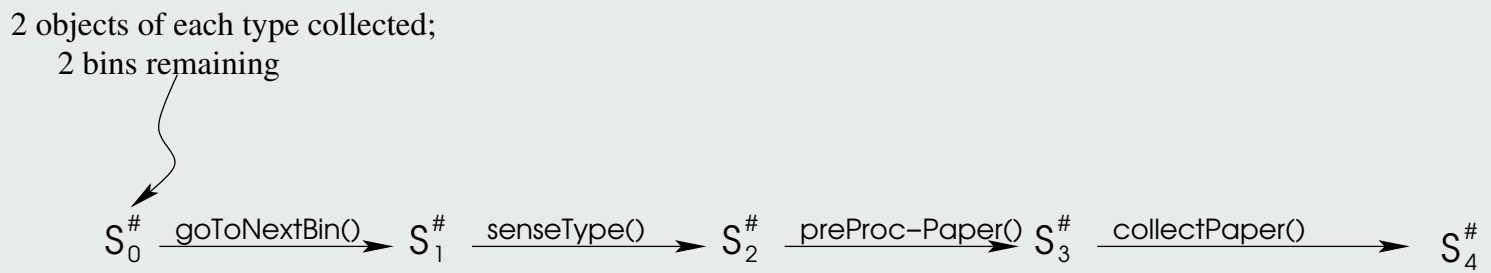
Summary of Action Application

- Draw-out action arguments prior to application
- Use focus and coerce to create cases for properties of drawn out of objects.
- Branches caused:
  - Classifiable, e.g $\#_R\{S\} > 1$
  - Unpredictable at planning-time, e.g. “object type=paper”

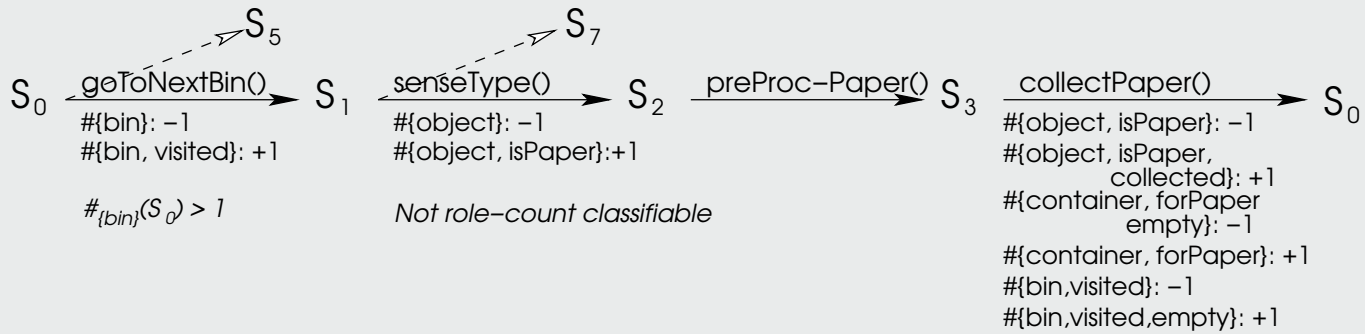
Plan Generalization

Use abstract structures to recognize loop invariants in example concrete plans.

Example Execution



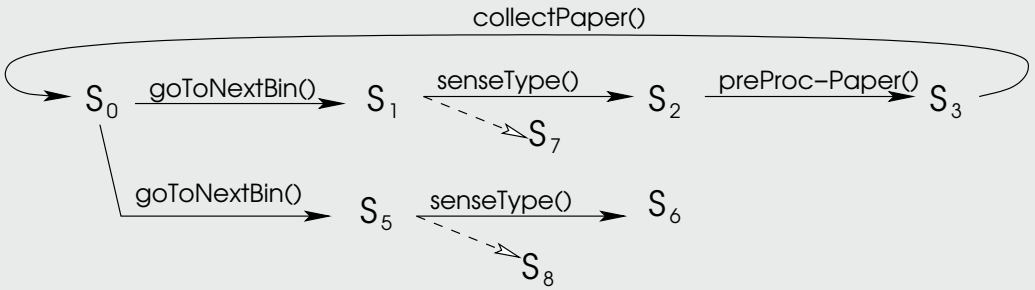
Tracing



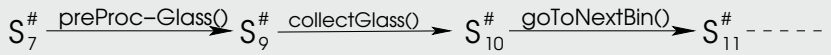
Developed for completely observable settings [Srivastava et al., 2008]

Merging Generalized Plans

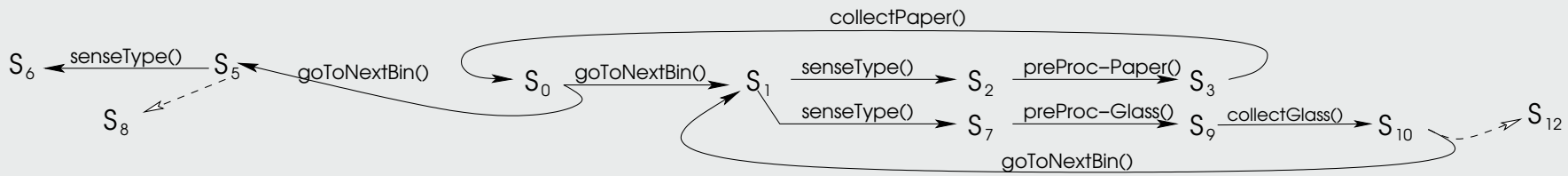
Find Loops



Plan for Unhandled Structure



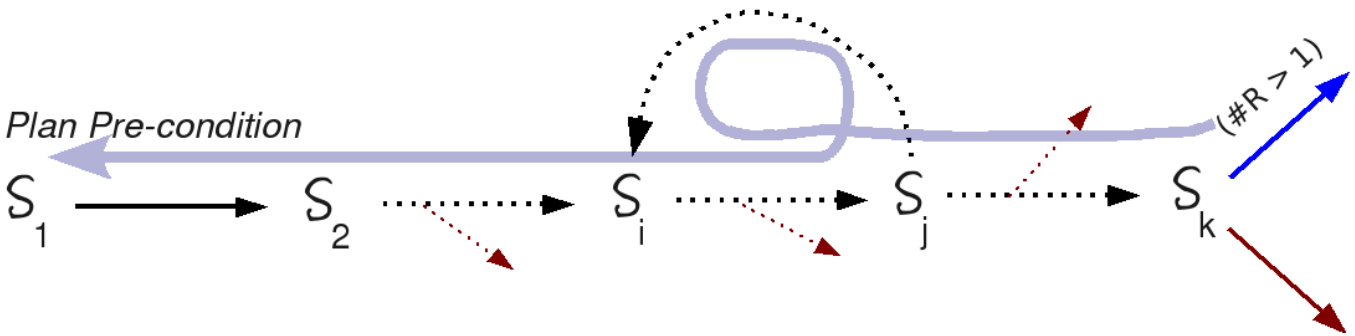
Generalize and Merge



- A single plan may not explore all possibilities.
- Construct problem instances from unsolved belief states.
- Solve them using classical planners.

Finding Preconditions

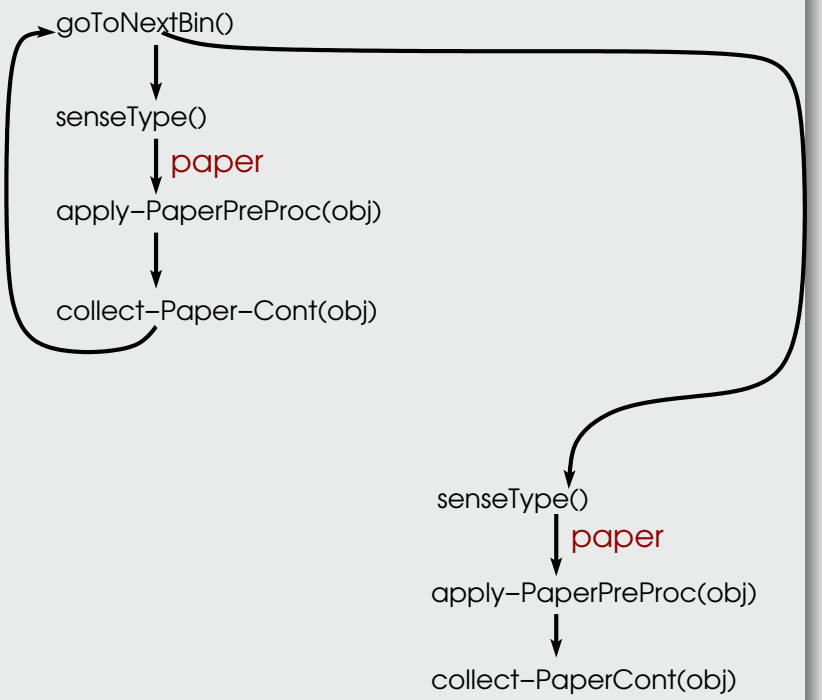
- Branches solve only *some* members of abstract structures.
- Classify branches using *role counts*; propagate backwards.
- Need for doing this constrains predicate update formulas.



Example Results

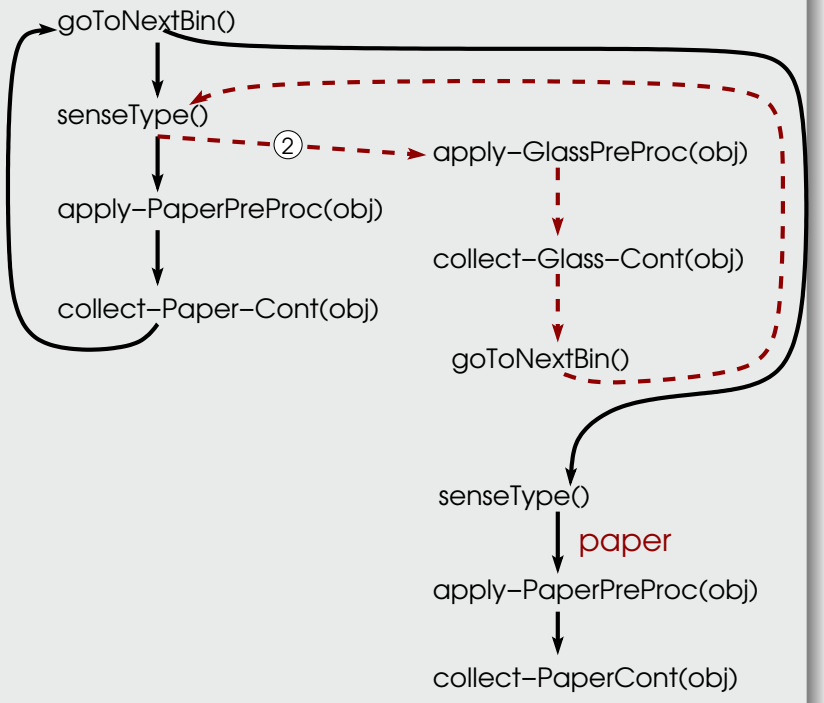
Initial: $p_0 = \|\{\text{paper, collected}\}\|; pc_0 = \|\{\text{empty,container,forPaper}\}\|;$
g_0, gc_0 : similar for glass; $b_0 = \|\{\text{bin}\}\|$

Loop 1



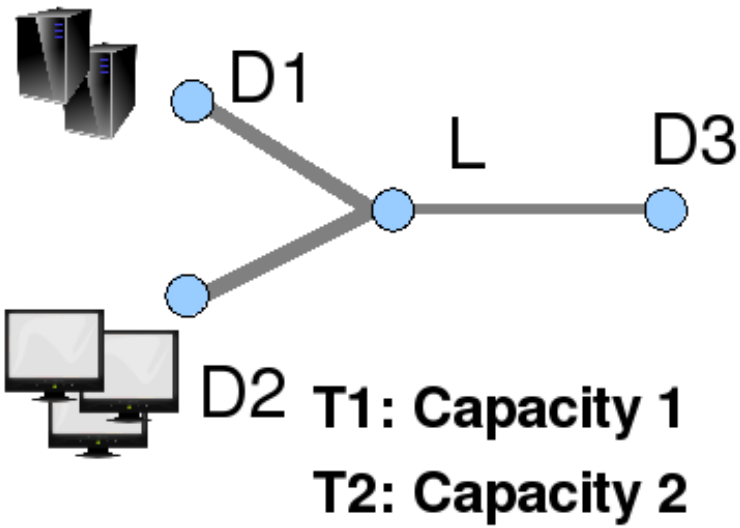
- Final: $p_0 + l_1; pc_0 - l_1; b_0 - l_1$
- Solves 1 instance out of every 2^n

Loops 1 & 2



- Final: $p_0 + l_1; pc_0 - l_1; g_0 + l_2; gc_0 - l_2; b_0 - l_1 - l_2$
- $2^{n-1} + 1$ out of every 2^n

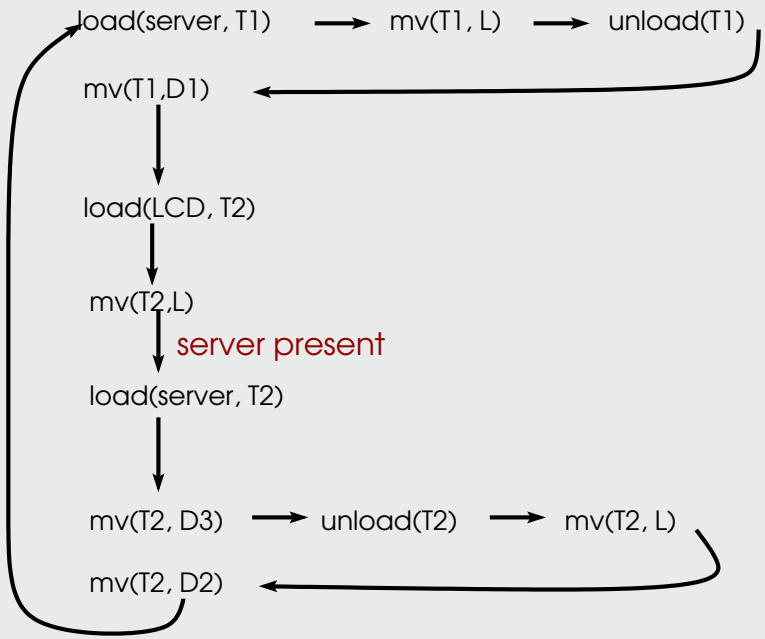
Transport Domain



Transport Domain: Results

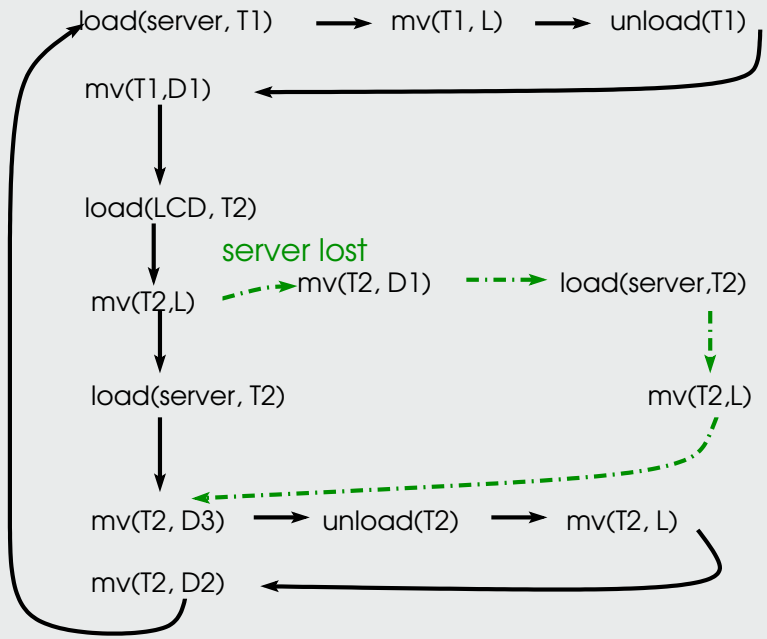
Initial undelivered counts:
$m_0 = \|\{\text{monitor, atD2}\}\|; s_0 = \|\{\text{server, atD1}\}\|$

Loop 1



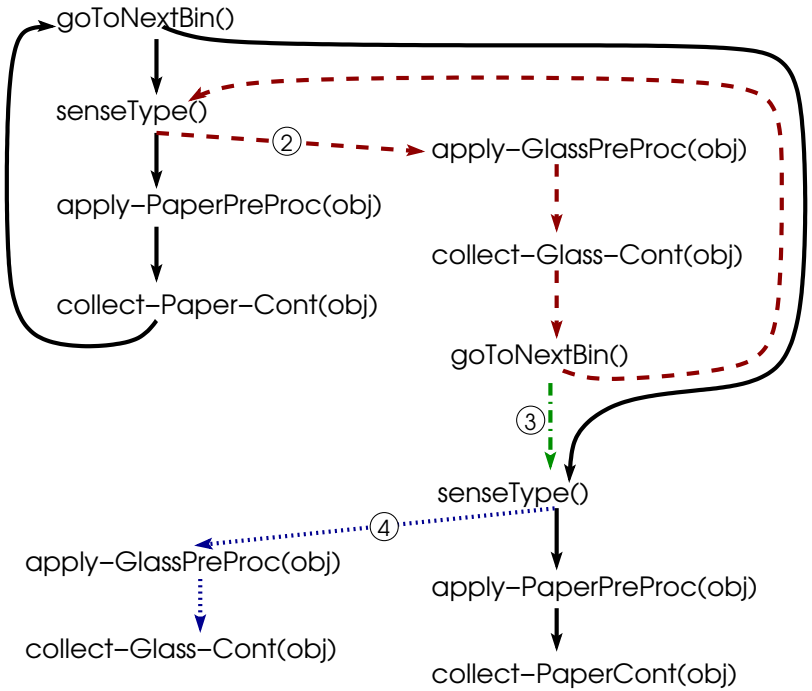
- Final: $m_o - l_1; s_0 - l_1$

Loops 1 & 2

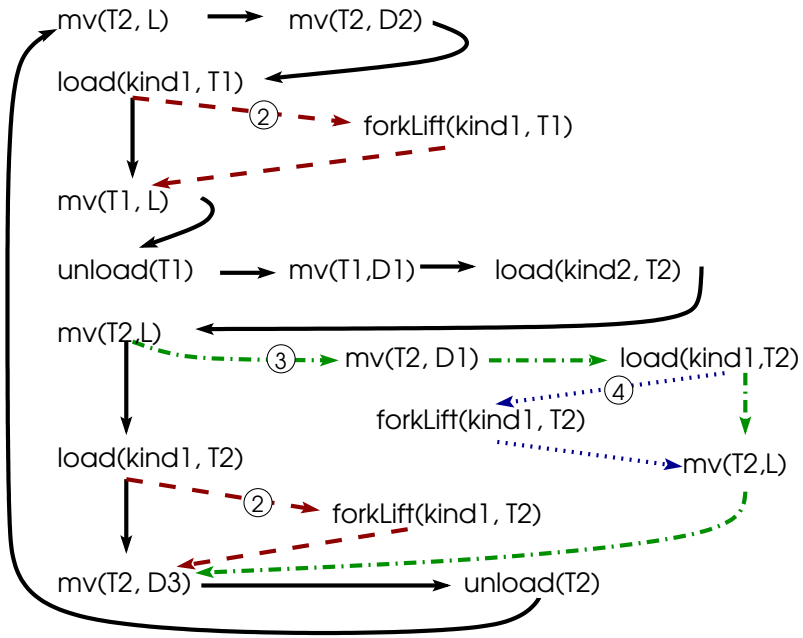


- Final: $m_0 - l_1; s_o - l_1 - k_1$

Solutions: Recycling and Transport



Recycling

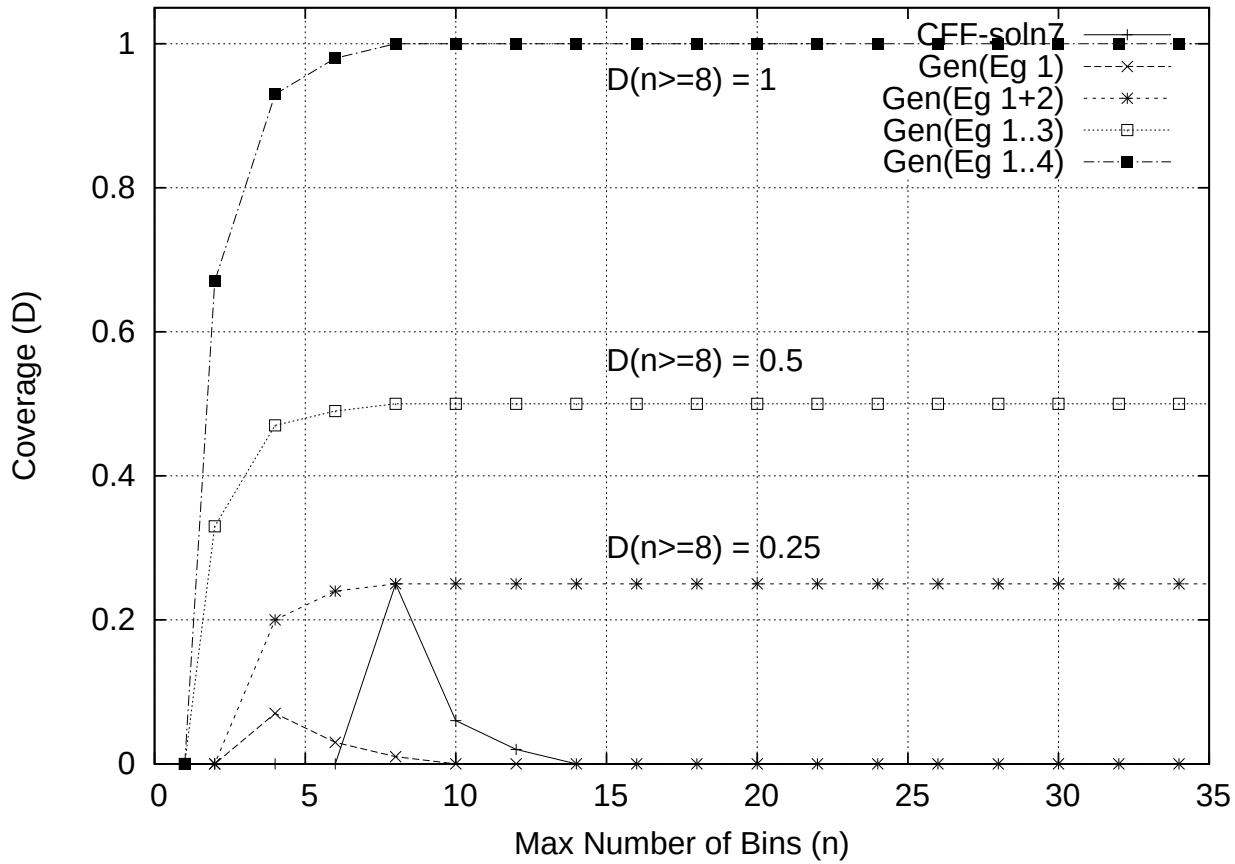


Transport

Conclusions

- An approach for representing unknown quantities for planning.
- Methods for finding generalized plans with branches and loops.
- Automatic computation of preconditions for many kinds of nested loops in a broad class of domains.

Example Results: Domain Coverage



$$D_\pi(n) = |\mathcal{S}_\pi(n)|/|\mathcal{T}(n)|$$

Merging Generalized Plans: Algorithm

Input: Existing plan Π, eg trace trace_i
Output: Extension of Π

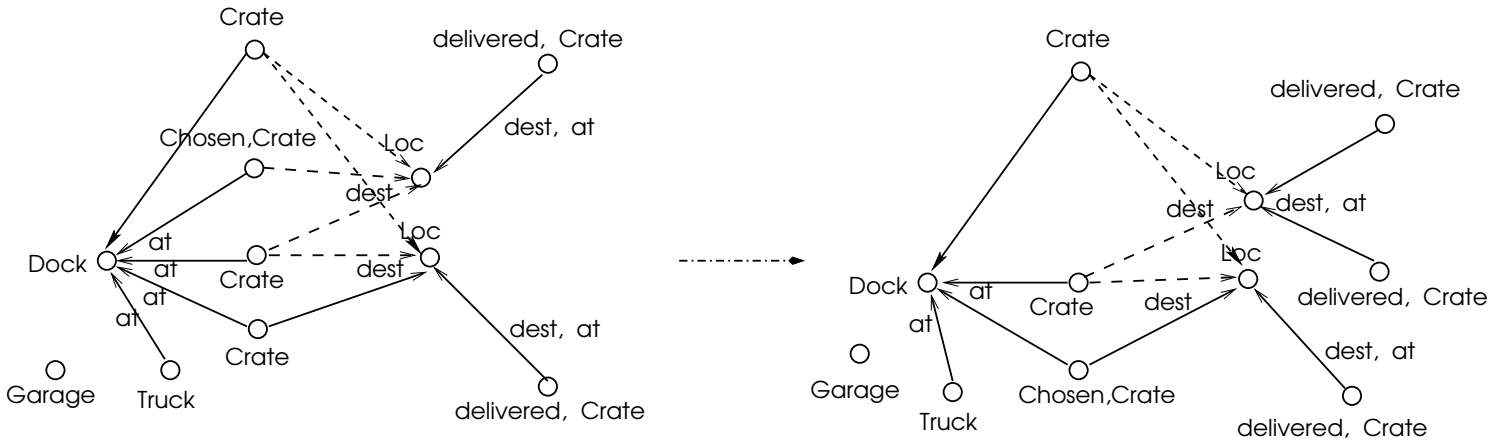
1 **if** $\Pi = \emptyset$ **then**
2 $\quad \Pi \leftarrow \text{trace}_i$
3 $\quad$ return Π
end
4 **repeat**
5 $\quad \text{mp}_\Pi, \text{mp}_t \leftarrow$ findMergePoint$(\Pi, \text{trace}_i, \text{bp}_\Pi, \text{bp}_t)$
6 $\quad$ **if** *mp_Π found and not first iteration* **then**
7 $\quad\quad$ attachEdges$(\Pi, \text{trace}_i, bp_t, mp_t, mp_\Pi, bp_\Pi)$
$\quad$ **end**
8 $\quad$ **if** *mp_Π found* **then**
9 $\quad\quad \text{bp}_\Pi, \text{bp}_t \leftarrow$ findBranchPoint$(\Pi, \text{trace}_i, \text{mp}_\Pi, \text{mp}_t)$
$\quad$ **end**
until *new bp_Π or mp_Π not found*
10 **return** Π

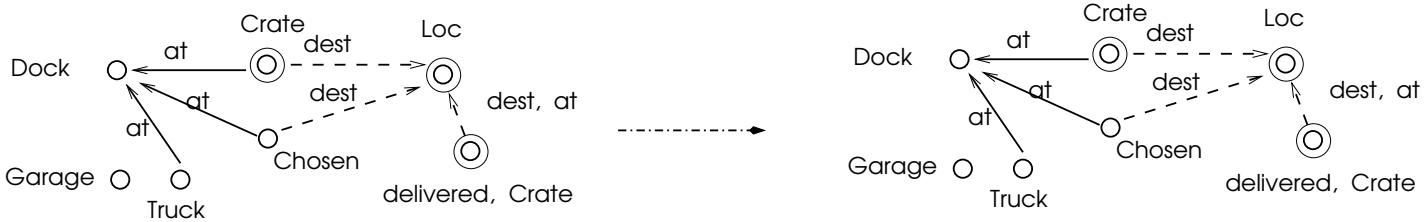
Algorithm 1: ARANDA-Merge

Related Work

- Plans with Loops
  - [Winner and Veloso, 2007]: no preconditions or sensing actions, but use partial ordering.
  - [Levesque, 2005]: single planning parameter, limited preconditions.
  - [Cimatti et al., 2003]: “hard” loops.
- Planning with unknown quantities:
  - [Milch et al., 2005]: action operators not provided.

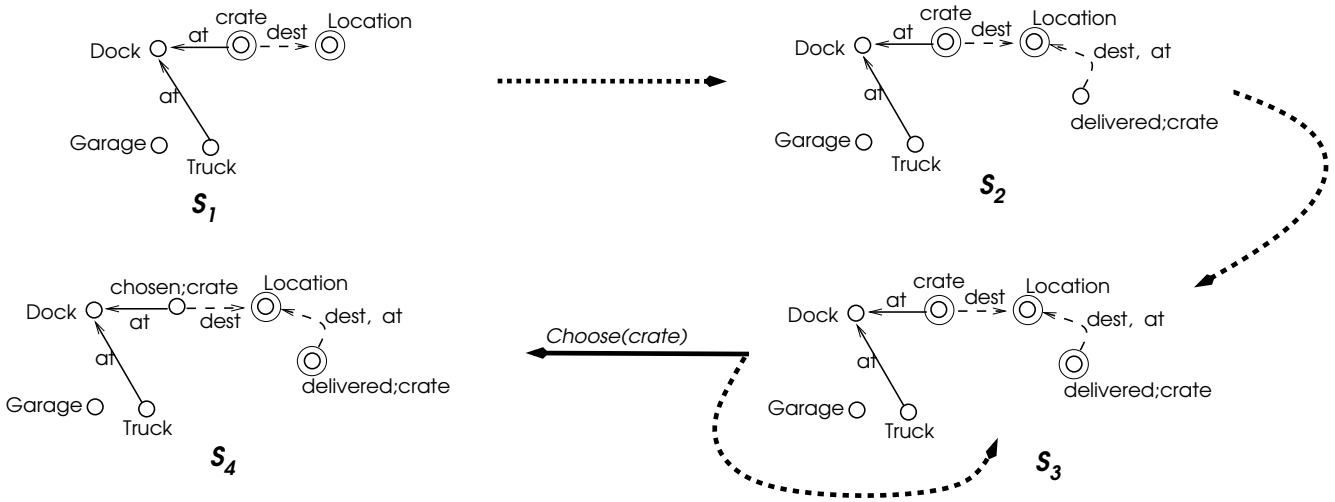
Plan Generalization: Example





= findDest(); Load(); setTarget(destLoc); Drive(); Unload(); Choose(Crate)

Plan Generalization: Example



References I

Cimatti, A., Pistore, M., Roveri, M., and Traverso, P. (2003).
Weak, strong, and strong cyclic planning via symbolic model checking.
Artif. Intell., 147(1-2):35–84.

Levesque, H. J. (2005).
Planning with loops.
In *Proc. of IJCAI*, pages 509–515.

Milch, B., Marthi, B., Russell, S. J., Sontag, D., Ong, D. L., and Kolobov, A. (2005).
Blog: Probabilistic models with unknown objects.
In *Proc. of IJCAI*, pages 1352–1359.

References II

Sagiv, M., Reps, T., and Wilhelm, R. (2002).
Parametric shape analysis via 3-valued logic.
ACM Transactions on Programming Languages and Systems, 24(3):217–298.

Srivastava, S., Immerman, N., and Zilberstein, S. (2008).
Learning generalized plans using abstract counting.
In *Proc. of AAAI*, pages 991–997.

Winner, E. and Veloso, M. (2007).
LoopDISTILL: Learning domain-specific planners from example plans.
In *Workshop on AI Planning and Learning, ICAPS*.