Universität Basel

Bachelor's Thesis Presentation

Schematic Invariant Synthesis Algorithm with Limited Grounding

Artifical Intelligence Research Group
Mario Tachikawa – mario.tachikawa@unibas.ch

17. June 2024

Introduction

1. ”Schematic Invariants by Reduction to Ground Invariants” by Jussi Rintanen

2. Implementation in Fast Downward

3. Evaluation with STRIPS benchmarks on sciCORE cluster

Task Example

```
(define (domain transport)
  (:requirements :typing :action-costs)
  (:types
        location locatable - object
        capacity-number - object
        vehicle package - locatable
  )
```

Example from: https://github.com/aibasel/downward-benchmarks

Task Example

```
(:predicates
    (road ?l1 ?l2 - location)
    (at ?x - locatable ?l - location)
    (in ?x - package ?v - vehicle)
    (capacity ?v - vehicle ?s1 - capacity-number)
    (capacity-predecessor ?s1 ?s2 - capacity-number)
)
```

Task Example

```
(:action drive
   :parameters (?v - vehicle ?l1 ?l2 - location)
   :precondition (and
       (at ?v ?l1)
       (road ?l1 ?l2)
   )
   :effect (and
       (not (at ?v ?l1))
       (at ?v ?l2)
   )
)
```

Task Example

```
(:objects
  city-loc-1 - location
  city-loc-2 - location
  city-loc-3 - location
  truck-1 - vehicle
  truck-2 - vehicle
  package-1 - package
  package-2 - package
  capacity-0 - capacity-number
  capacity-1 - capacity-number
  capacity-2 - capacity-number
)
```

Task Example

```
(:init
  (capacity-predecessor capacity-0 capacity-1)
  (capacity-predecessor capacity-1 capacity-2)
  (at package-1 city-loc-3)
  (at package-2 city-loc-3)
  (at truck-1 city-loc-3)
  (capacity truck-1 capacity-2)
  (at truck-2 city-loc-1)
  (capacity truck-2 capacity-2)
)
```

Invariant

- Formula that is true in all reachable states
- Formulas of the form:

$$
\begin{aligned}
\chi &\implies l_1 \\
\chi &\implies (l_1 \vee l_2)
\end{aligned}
$$

- Examples:
- Schematic invariant candidate:

$$(v_1 \neq v_2) \implies (\neg in(x, v_1) \vee \neg in(x, v_2))$$

- Ground invariant candidate:

$$\neg in(\text{package-1}, \text{truck-1}) \vee \neg in(\text{package-1}, \text{truck-2})$$

Schematic Invariant Synthesis Algorithm

Algorithm 1 Schematic Invariant Synthesis Algorithm

$C_s :=$ schematic formulas true in the initial state
$A_s :=$ schematic actions
$C :=$ all ground instances of C_s
    by instantiating all schematic formulas in C_s using limited grounding
$A :=$ all grounded actions
    by instantiating all schematic actions in A_s using limited grounding
repeat
    $C_0 := C$
    for each $a \in A$ and $c \in C$ **do**
        if $C_0 \cup \{regr_a(\neg c)\} \in$ SAT for some c **then**
            $C := (C \setminus \{c\}) \cup$ weaken(c)
        end if
    end for
until $C = C_0$
$I_s :=$ all schematic invariants
    extracted from the found ground invariants in C
return I_s

Implementation

- Translation of PDDL task into Finite Domain Representation (FDR) task:
  1. Normalization
  2. Invariant synthesis
  3. Generation of mutex groups
  4. FDR task generation
- Maximum clique enumeration algorithm by Tomita
- Difficulty: initial schematic invariant candidate set

Initial Schematic Invariant Candidate Set

- Goal: Strongest possible candidates

- Procedure:

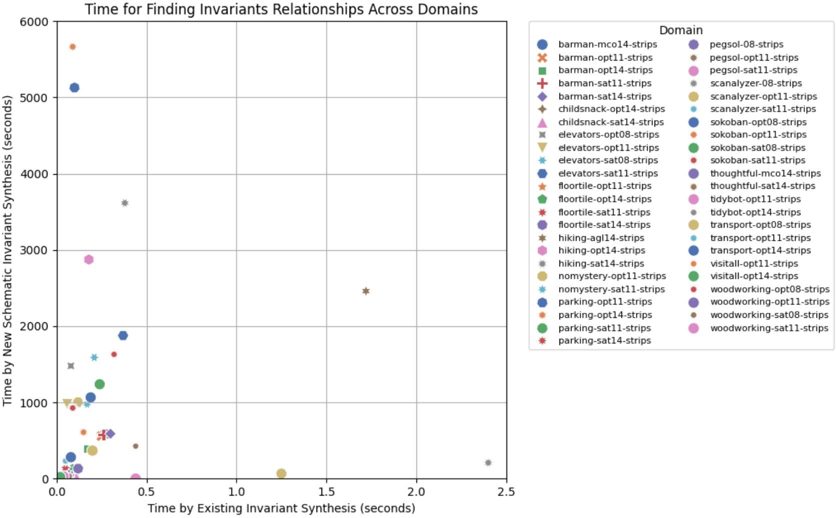
1. Single literal invariant candidate:
   - Example: $\neg in(?x, ?v)$
2. Weaker forms:
   - Inequality:
     $(?x1 \neq ?x2) \implies \neg at(?x1, ?l1) \vee \neg at(?x2, ?l2)$
   - Equality:
     $\neg at(?x, ?l1) \vee \neg at(?x, ?l2)$
   - Add literal:
     $\neg at(?x1, ?l1) \vee \neg at(?x2, ?l2)$

Evaluation

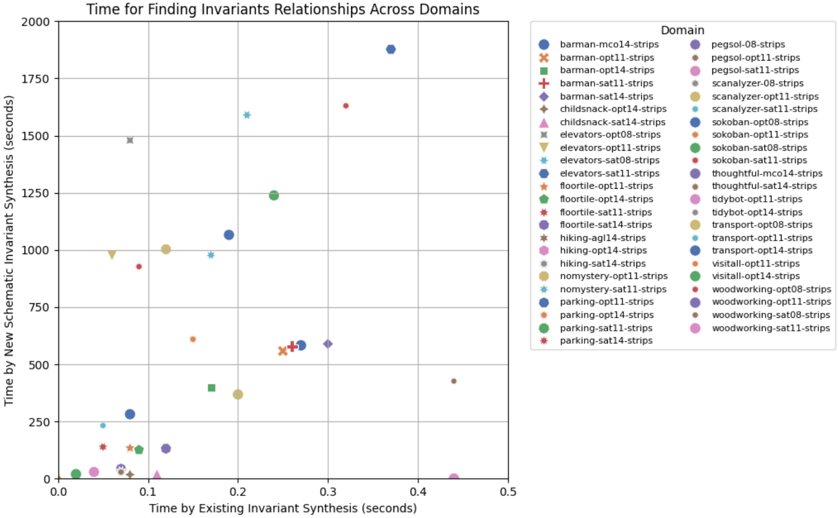
- Comparison of planner using existing invariant synthesis

- Evaluated Metrics:
  1. Time and Memory Errors
  2. Time Performance for Finding Invariants
  3. Search Time Performance
  4. Plan Validity
  5. Finite Domain Variable Structure
     - Number of Variables
     - Average Number of possible Values per Variable

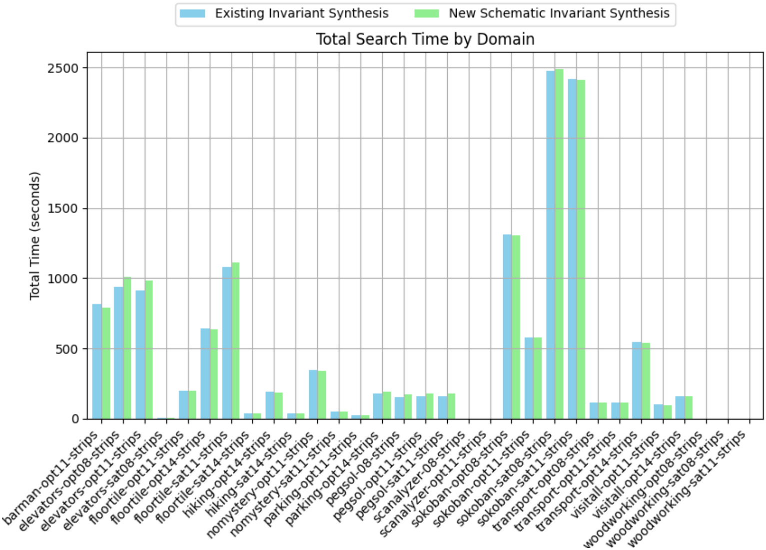
Time Performance for Finding Invariants



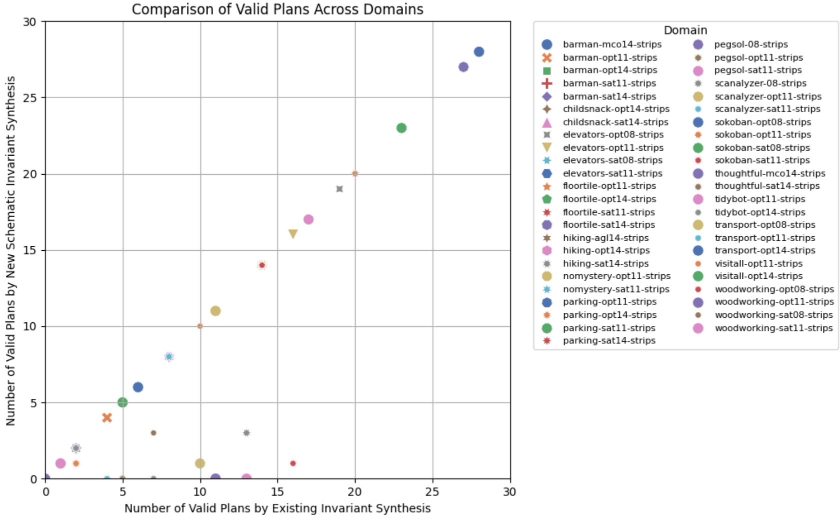
Time Performance for Finding Invariants



Search Time Performance



Plan Validity



Conclusion

- Implementation too slow
- More suitable data structures
- Similar impact on search

- Implementation and integration: challenging and time-consuming

Questions

Additional Slide: Limited Grounding

- Grounding function: $D(t) =$ set of objects for type t

- Limited grounding function $D'(t)$, with the following characteristics:

$$D'(t) = D(t)$$

or

1. $D'(t) \subset D(t)$
2. $|D'(t)| \geqslant L_t^N(A, P)$
3. $D'(t_0) \subset D'(t_1)$ iff $D(t_0) \subset D(t_1)$ for all $\{\{t_0, t_1\} | \, t_0, t_1 \in T\}$

- Lower bound number:

$$L_t^N(A, P) = max(max_{a \in A} prms_t(a), max_{p \in P} prms_t(p)) + (N - 1) * (max_{p \in P} prms_t(p))$$