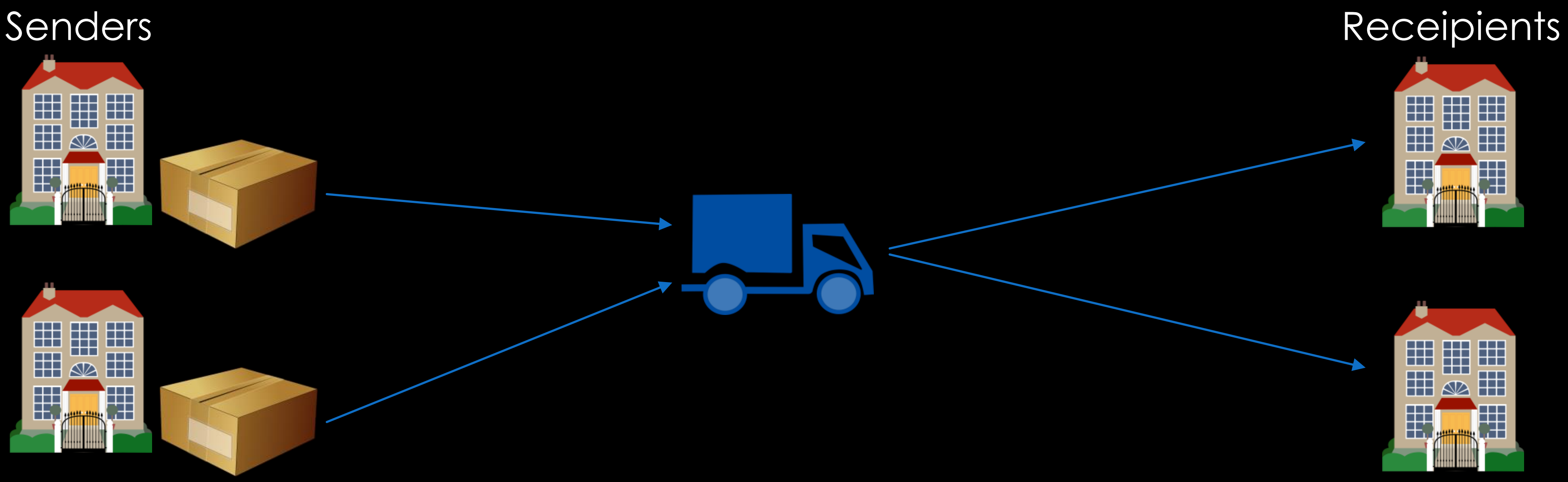
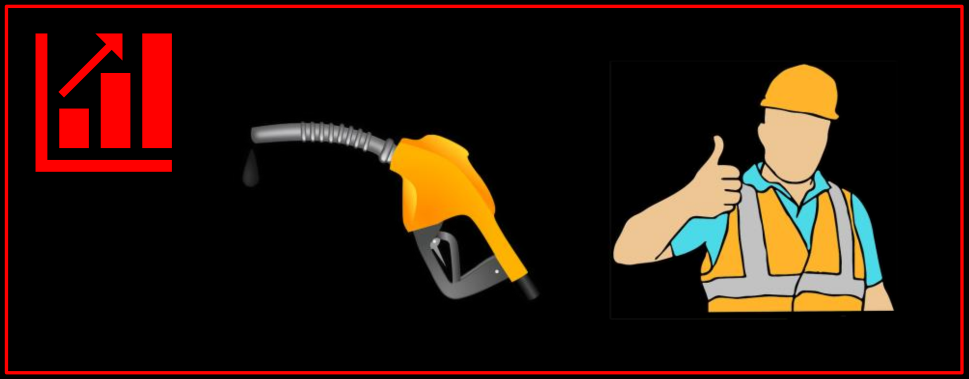
Generation of Domain Abstractions using Counterexample-Guided Abstraction Refinement

Bachelor‘s Thesis – Raphael Kreft

Cost-Optimal Classical Planning





SAS$^+$ Planning Formalism

Planning task $\Pi = < V, s_0, G, A >$ with

- A set of state variables V where each $v \in V$ is associated with a finite, non-empty domain.
  - State = total assignment for V
- A state s_0 which is called the initial state
- A variable assignment G which denotes the goal conditions
- A finite set of actions A, where each action $a \in \mathrm{A}$ is associated with:
  - two variable assignments, namely Effects $\mathbf{eff}(\boldsymbol{a})$ and Preconditions $\mathbf{pre}(\boldsymbol{a})$
  - Non negative costs $\boldsymbol{cost}(\boldsymbol{a}) \in \mathbb{N}_{\mathbf{0}}$

Example:

Position of Package A

Position of Truck

SAS$^+$ Planning Formalism

Planning task $\Pi =< V, s_0, G, A >$ with

- A set of state variables V where each $v \in V$ is associated with a finite, non-empty domain.
  - State = total assignment for V
- A state s_0 which is called the initial state
- A variable assignment G which denotes the goal conditions
- A finite set of actions A, where each action $a \in \mathrm{A}$ is associated with:
  - two variable assignments, namely Effects $\mathbf{eff}(\boldsymbol{a})$ and Preconditions $\mathbf{pre}(\boldsymbol{a})$
  - Non negative costs $\boldsymbol{cost}(\boldsymbol{a}) \in \mathbb{N}_\mathbf{0}$

Example:

All packages are at start location

Trucks are in their base

SAS^+ Planning Formalism

Planning task $\Pi = < V, s_0, G, A >$ with

- A set of state variables V where each $v \in V$ is associated with a finite, non-empty domain.
    - State = total assignment for V
- A state s_0 which is called the initial state
- A variable assignment G which denotes the goal conditions
- A finite set of actions A, where each action $a \in \mathrm{A}$ is associated with:
    - two variable assignments, namely Effects $\mathbf{eff}(\boldsymbol{a})$ and Preconditions $\mathbf{pre}(\boldsymbol{a})$
    - Non negative costs $\boldsymbol{cost}(\boldsymbol{a}) \in \mathbb{N}_{\mathbf{0}}$

Example:

All packages are at their final destination

It does not care where the trucks are

SAS$^+$ Planning Formalism

Planning task $\Pi =< V, s_0, G, A >$ with

- A set of state variables V where each $v \in V$ is associated with a finite, non-empty domain.
  - State = total assignment for V
- A state s_0 which is called the initial state
- A variable assignment G which denotes the goal conditions
- A finite set of actions A, where each action $a \in A$ is associated with:
  - two variable assignments, namely Effects $\mathbf{eff}(\boldsymbol{a})$ and Preconditions $\mathbf{pre}(\boldsymbol{a})$
  - Non negative costs $\boldsymbol{cost}(\boldsymbol{a}) \in \mathbb{N}_0$

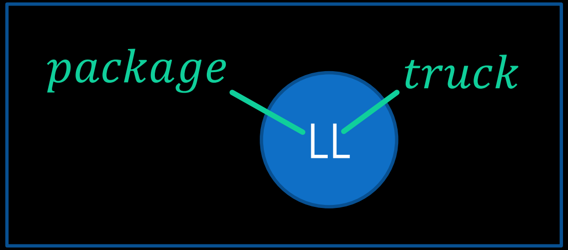
Example:

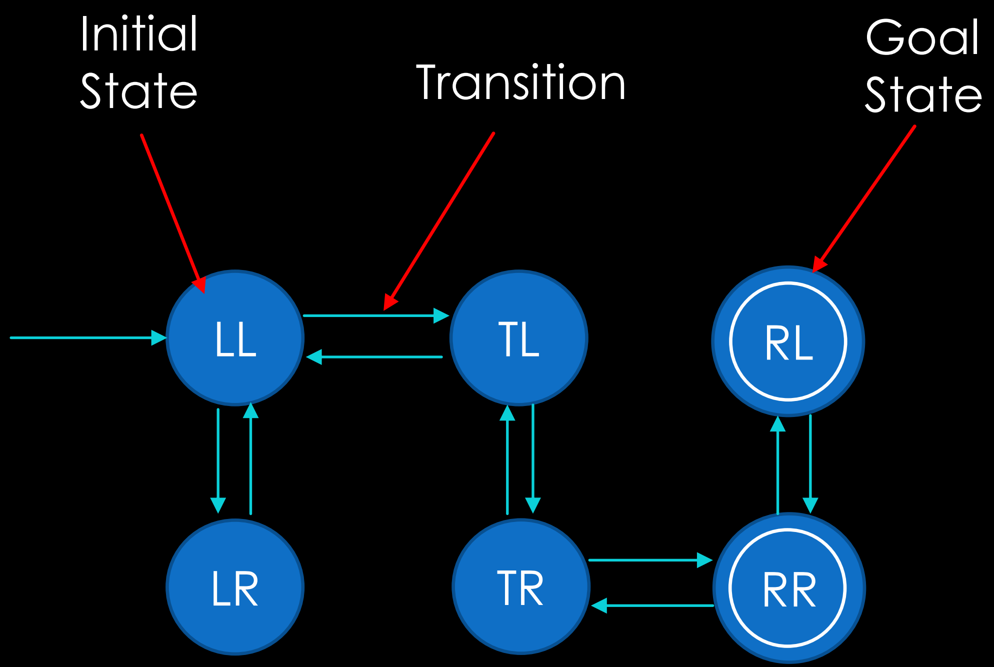
Action **load(packageA, truckA)**

- **Preconditions:** packageA and truckA must be in same location
- **Effects:** position of packageA is set to truckA

Example Statespace

$V = \{package, truck\},$
$\mathrm{dom}(package) = \{\mathrm{L,R,T}\},$
$\mathrm{dom}(truck) = \{L, R\}$
$A = \{load, unload, move\}$



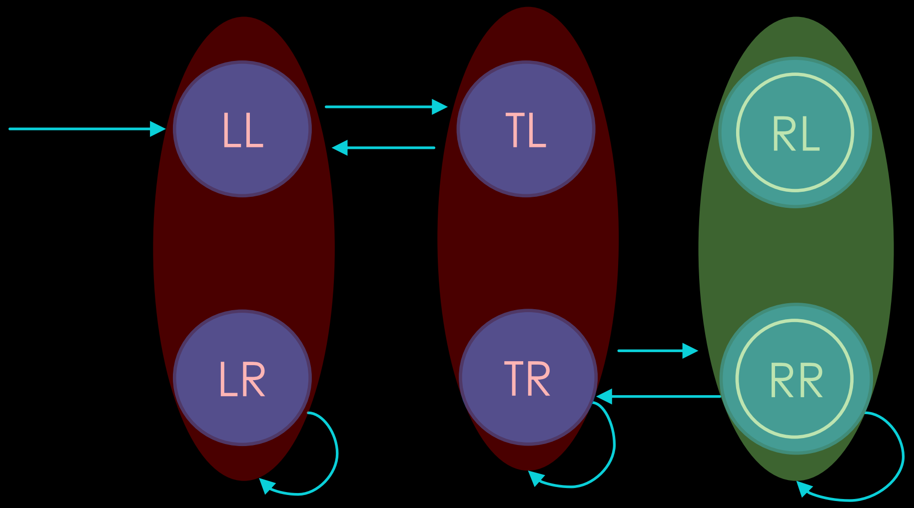


Heuristics

$$h: S \rightarrow \mathbb{R}_0^+ \cup \{\infty\}$$

- Estimate optimal goal distance for all states
- „Guides“ a search algorithm
- Handcrafted possible: ex. Manhatten distance
- Many methods to derive automatically

Abstraction Heuristics



$$h^{\alpha}(\{package \rightarrow L, truck \rightarrow L\}) = 2$$

Generation of Domain Abstractions using Counterexample-Guided Abstraction Refinement

Bachelor‘s Thesis – Raphael Kreft

- Domain Abstractions
- Counterexample-Guided Abstraction Refinement
- Constructing Domain Abstractions with CEGAR
- Evaluation and Comparison
- Conclusion and Future Work

Domain Abstractions

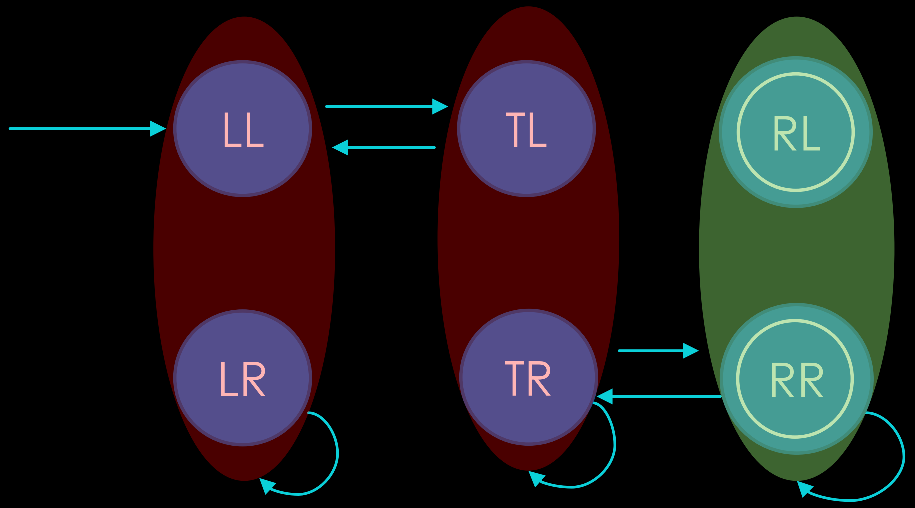
And other abstraction classes

Abstraction Classes

Projections

Domain Abstractions

Cartesian Abstractions



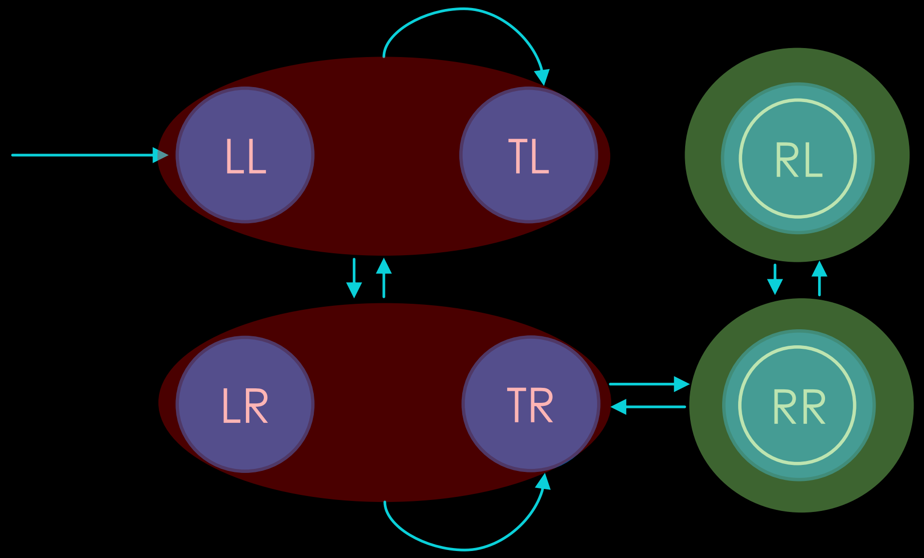
projection on variable $package$

Abstraction Classes

Projections

Domain Abstractions

Cartesian Abstractions



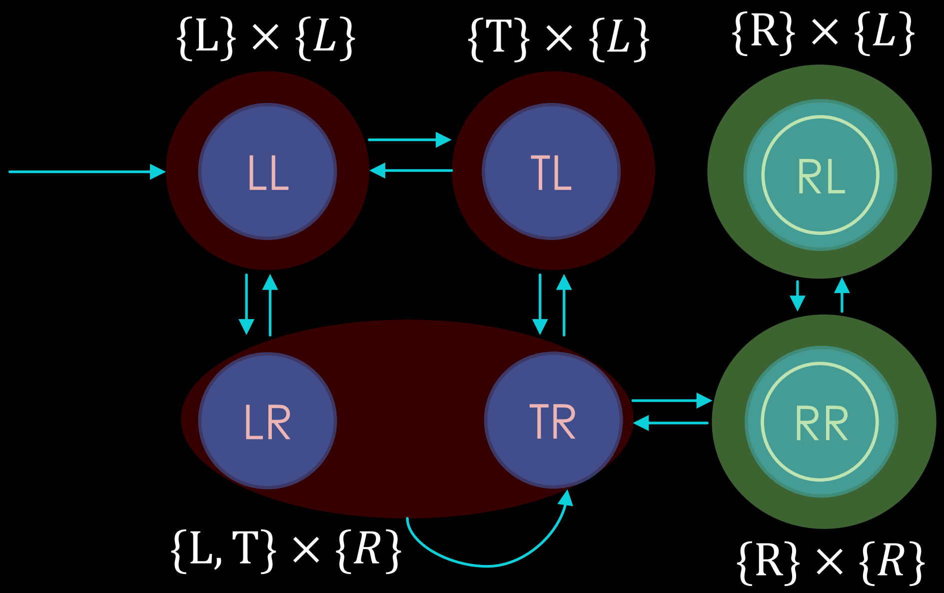
$\mathrm{dom}(package) = \{\mathrm{L}, \mathrm{R}, \mathrm{T}\}$ $\mathrm{dom}(truck) = \{L, R\}$

Abstraction Classes

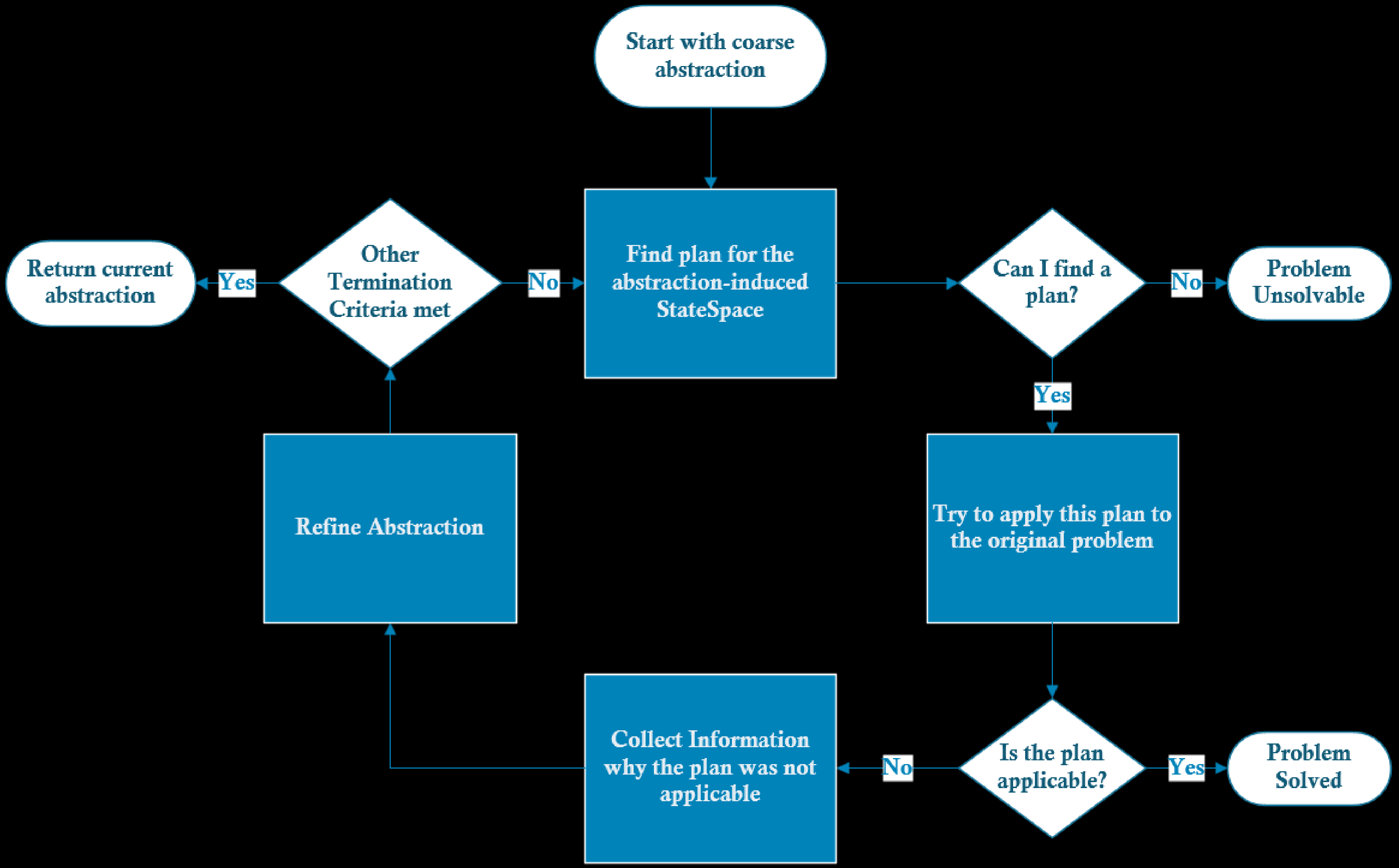
Projections

Domain Abstractions

Cartesian Abstractions

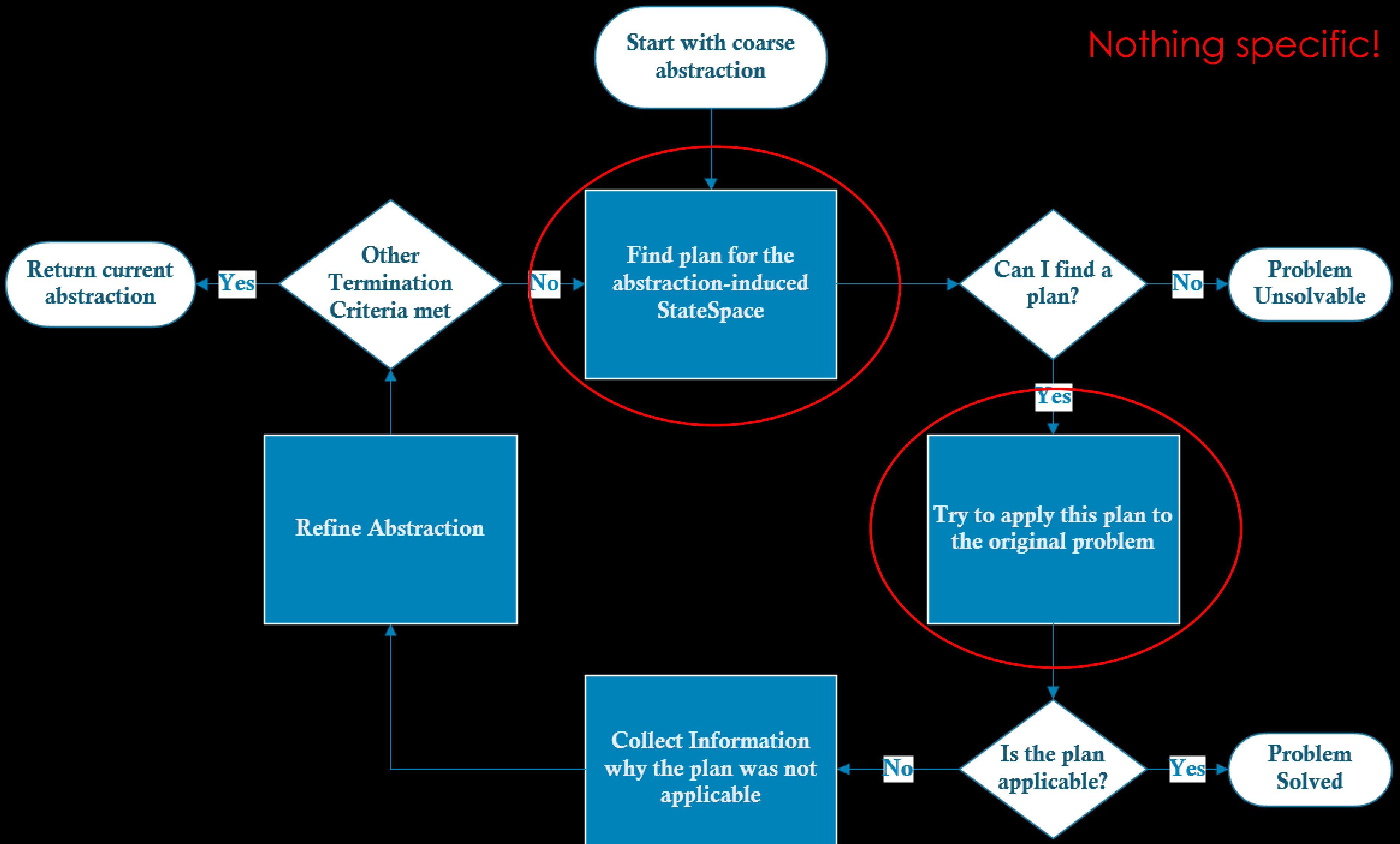


Counterexample-guided Abstraction Refinement


Start with coarse abstraction
Find plan for the abstraction-induced StateSpace
Can I find a plan?
No
Problem Unsolvable
Yes
Try to apply this plan to the original problem
Is the plan applicable?
Yes
Problem Solved
No
Collect Information why the plan was not applicable
Refine Abstraction
Other Termination Criteria met
No
Yes
Return current abstraction

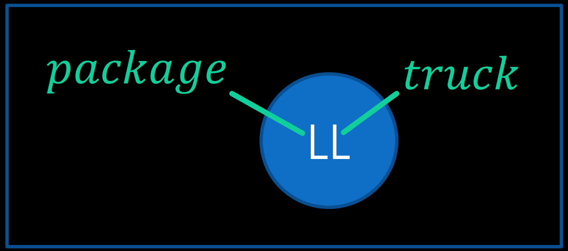
Constructing Domain Abstraction using CEGAR

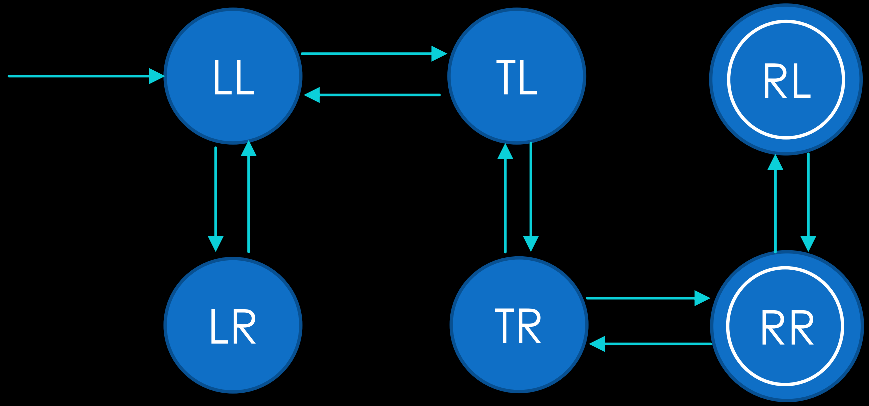
Motivation, Algorithm and Parameters


Start with coarse abstraction
Find plan for the abstraction-induced StateSpace
Can I find a plan?
No
Problem Unsolvable
Yes
Try to apply this plan to the original problem
Is the plan applicable?
Yes
Problem Solved
No
Collect Information why the plan was not applicable
Refine Abstraction
Other Termination Criteria met
No
Yes
Return current abstraction
Nothing specific!

Flaws

$V = \{package, truck\},$
$\quad \mathrm{dom}(package) = \{\mathrm{L,R,T}\},$
$\quad \mathrm{dom}(truck) = \{L, R\}$
$s_0 = \{truck \rightarrow L, package \rightarrow L\}$
$G = \{package \rightarrow R\}$
$A = \{load, unload, move\}$

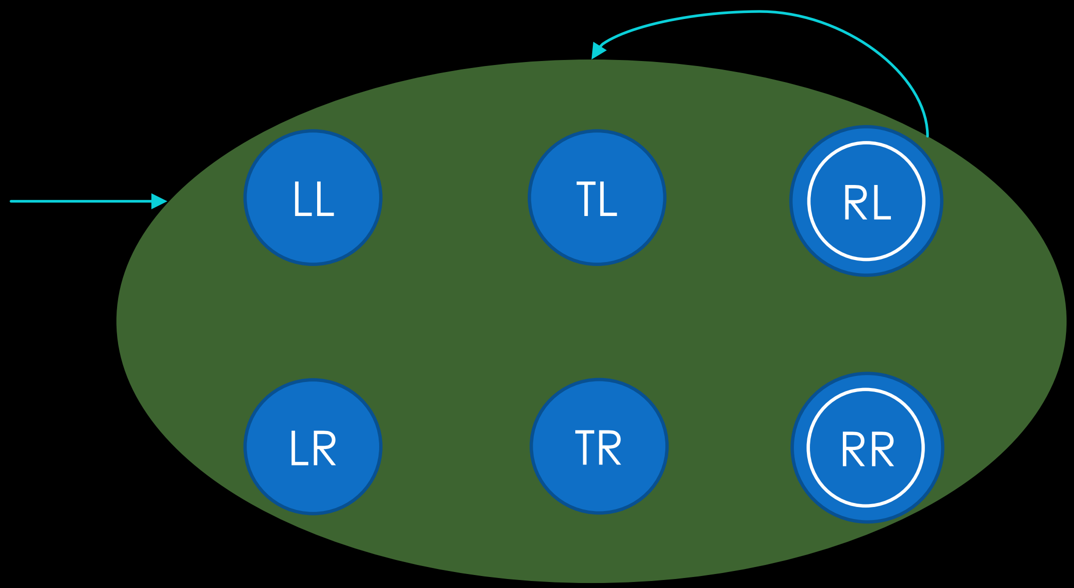




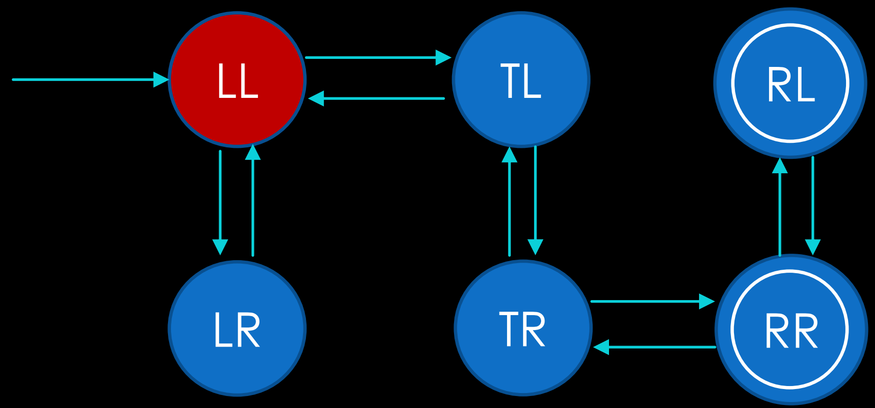
Start with coarse abstraction

Initial Domain Abstraction:

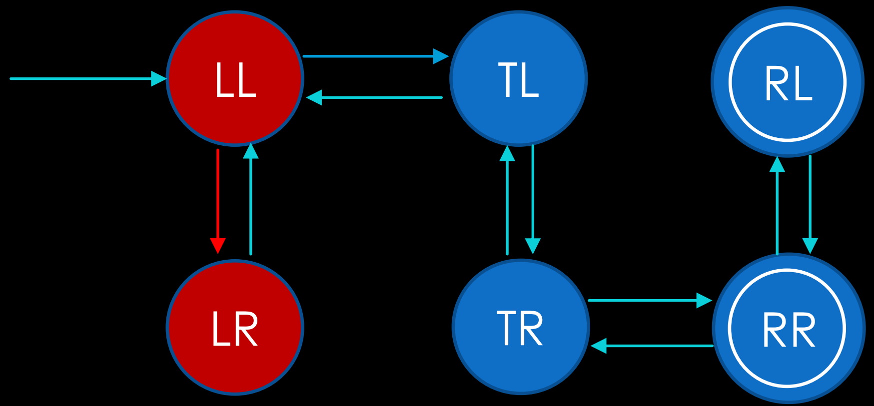
$\mathrm{dom}(package) = \{\mathrm{L}, \mathrm{R}, \mathrm{T}\}$ $\mathrm{dom}(truck) = \{L, R\}$



Flaws



$\pi = \langle\rangle$



$\pi = \langle move, \color{red}{load} \rangle$

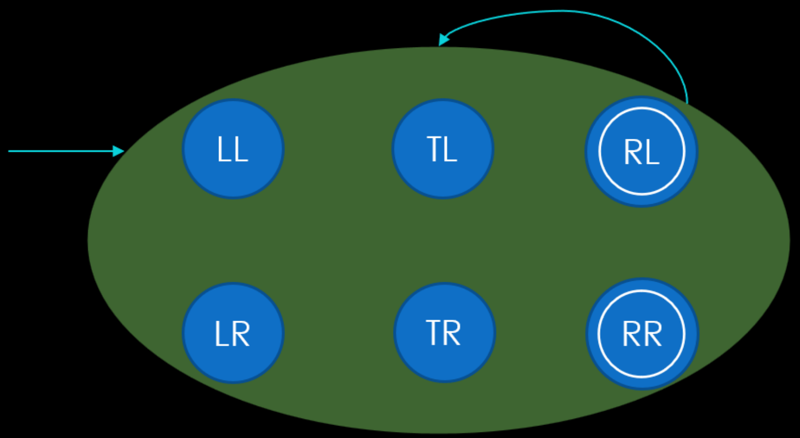
Let s be the state where flaw occurred:

Precondition Flaw:	$f = pre(a) \setminus \{s\}$
Goal Flaw:	$f = G \setminus \{s\}$

Refine Abstraction

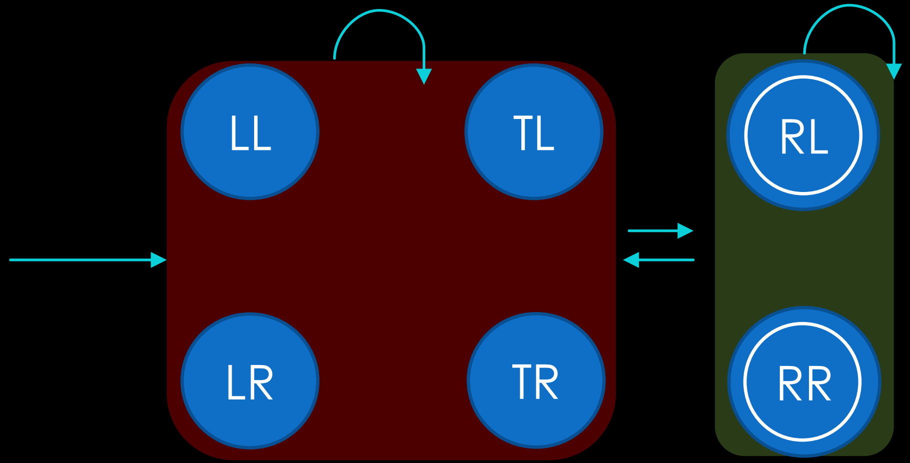
Old Abstraction:

$\mathrm{dom}(package) = \{\mathrm{L}, \mathrm{R}, \mathrm{T}\}$
$\mathrm{dom}(truck) = \{L, R\}$



New Abstraction:

$\mathrm{dom}(package) = \{\mathrm{L}, \mathrm{R}, \mathrm{T}\}$
$\mathrm{dom}(truck) = \{L, R\}$

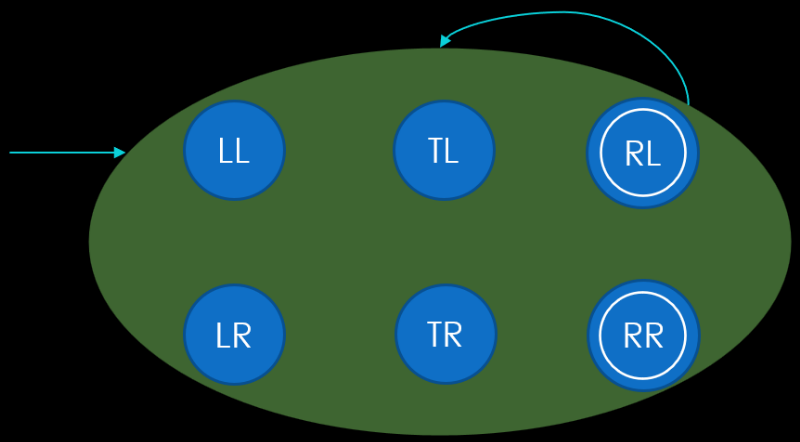


Adjustable Parameters

1. **Initial Abstraction Selection**

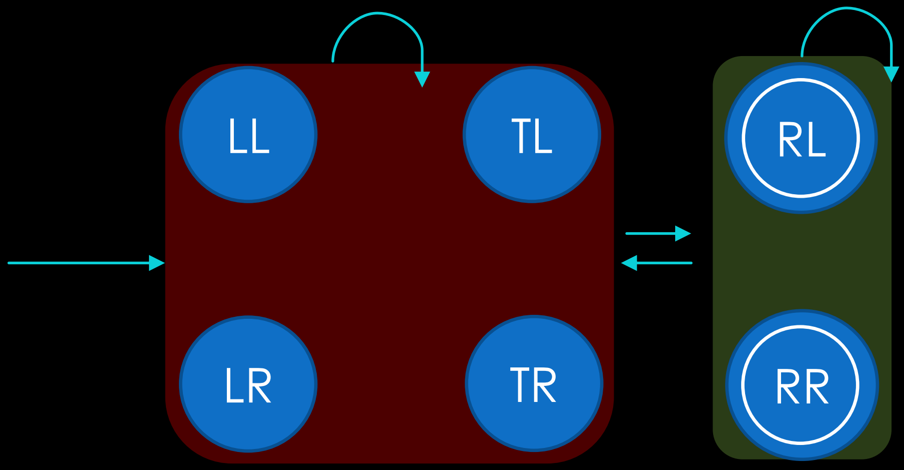
Most Coarse Abstraction

$\mathrm{dom}(package) = \{\mathrm{L, R, T}\}$
$\mathrm{dom}(truck) = \{L, R\}$



Goal Facts initially splitted

$\mathrm{dom}(package) = \{\mathrm{L}, \mathrm{R}, \mathrm{T}\}$
$\mathrm{dom}(truck) = \{L, R\}$



Adjustable Parameters

2. **How many facts to split**

Given a flaw with multiple missed assignments: $f = \{v_1 \rightarrow 1, v_4 \rightarrow 3\}$

<u>Use one FactPair</u>

- Choose one fact pair of flaw f
  ex. $v_1 \rightarrow 1$
  1. Uniformly at Random
  2. Max refined domain
  3. Least refined domain

<u>Use all FactPairs</u>

- Use all fact pairs for refinement
- Implementation: Split as many as possible

Adjustable Parameters

3. How to split according to one assignment

Given the flaw from example before: $f = \{package \rightarrow R\}$

<u>Single Value Split</u>

$$\mathrm{dom}(package) = \{\mathrm{L}, \mathrm{R}, \mathrm{T}\}$$
$$\mathrm{dom}(truck) = \{L, R\}$$

- Only move missed value(R) in a new equivalence class
- This was the method used in the Refinement Example

<u>Uniform Random Split</u>

$$\mathrm{dom}(package) = \{\mathrm{L}, \mathrm{R}, \mathrm{T}\}$$
$$\mathrm{dom}(truck) = \{L, R\}$$

- Additionally missed-value(R), choose other values from same equivalence class uniformly at random
- Move 50% of old equivalence class to new one

Adjustable Parameters

4. Abstraction Size Limit

- Equals the product of the number of equivalence classes for each variable domain

- Influences the effort of:
  - Refinement Loop (Find solution in abstract state space)
  - Obtain Heuristic Values

Adjustable Parameters

Request heuristic value for state $\boldsymbol{s}$

Obtain abstract state $\boldsymbol{s_\alpha}$ corresponding to $\boldsymbol{s}$

Obtain abstract goal distance

1. On Demand
2. Precomputation

Evaluation

Best Configurations and Comparison to others

Setup

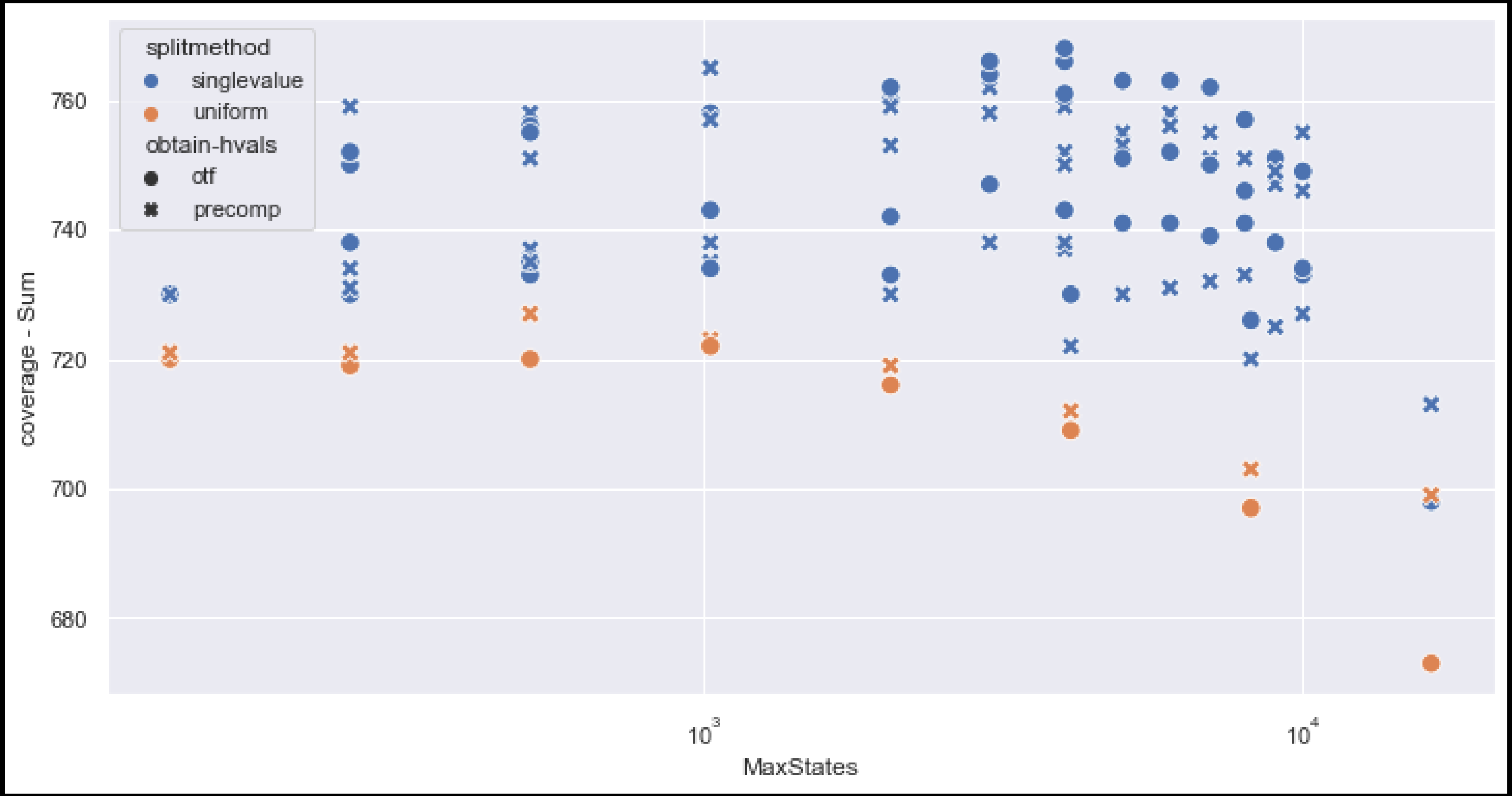
- Algorithms Implementated in Planning System Fast Downward
- Setup Experiments with Downward Lab
- Experiments performed on SciCore(Infai2 Cluster)
- Set of 1827 tasks from 65 different problem domains

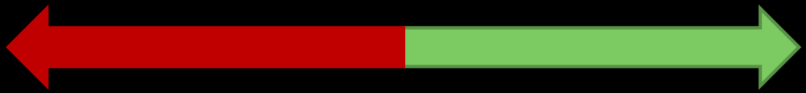
For Each Task

- Overall Time Limit: 30min
- Overall Memory Limit: 3.5GB

→ **Performance Evaluation per Parameter!**

Maximum Abstraction Size



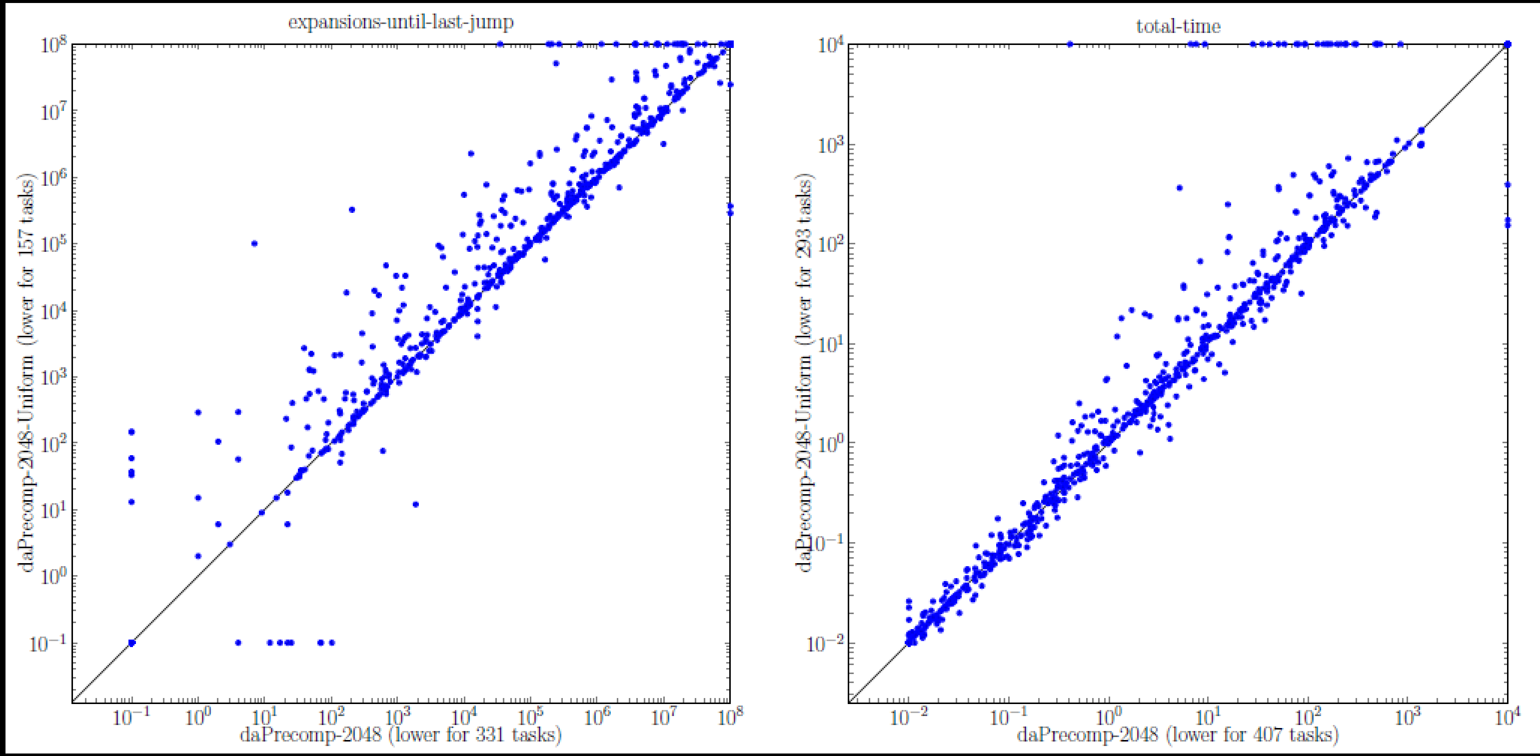


Lower
- Computational effort
- Accuracy

Higher
- Computational effort
- Accuracy

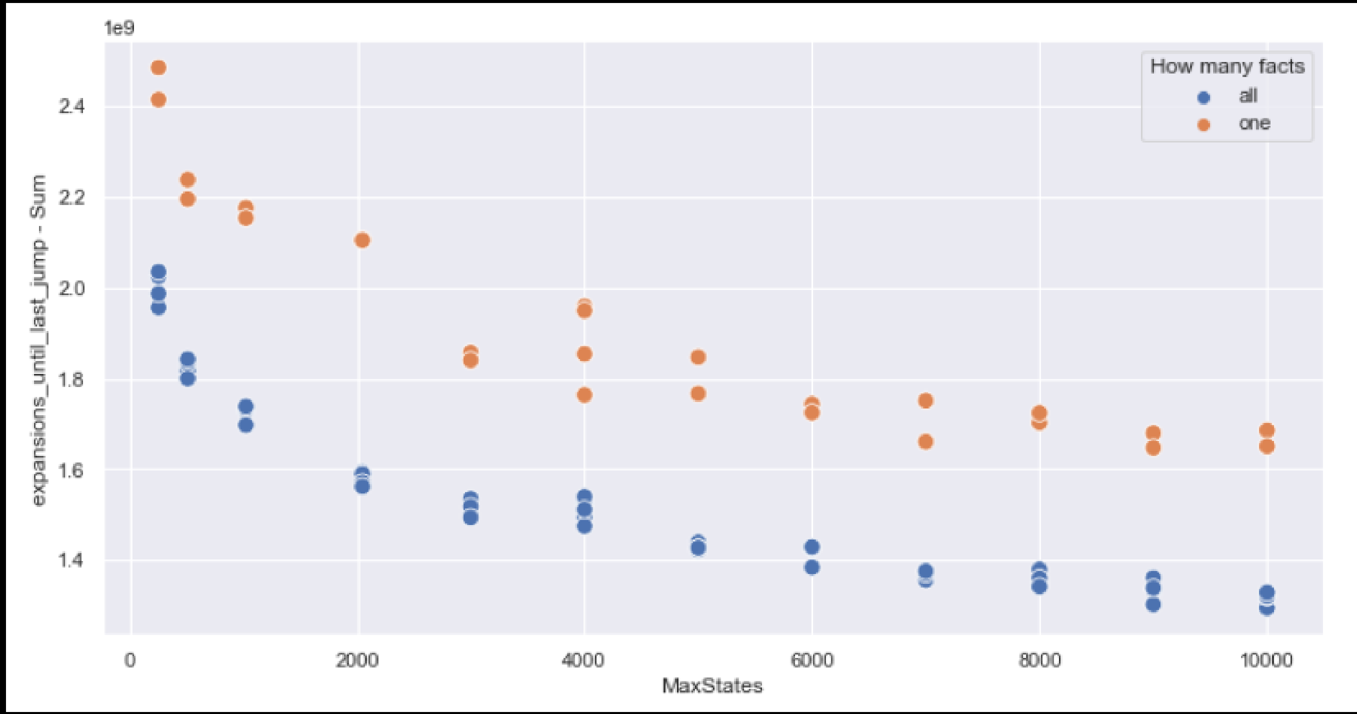
Split Method

- Single Value Split is superior in terms of covergae, time and informativeness
- Configurations using Uniform Random Split performed worse in nearly all cases



How many FactPairs to split

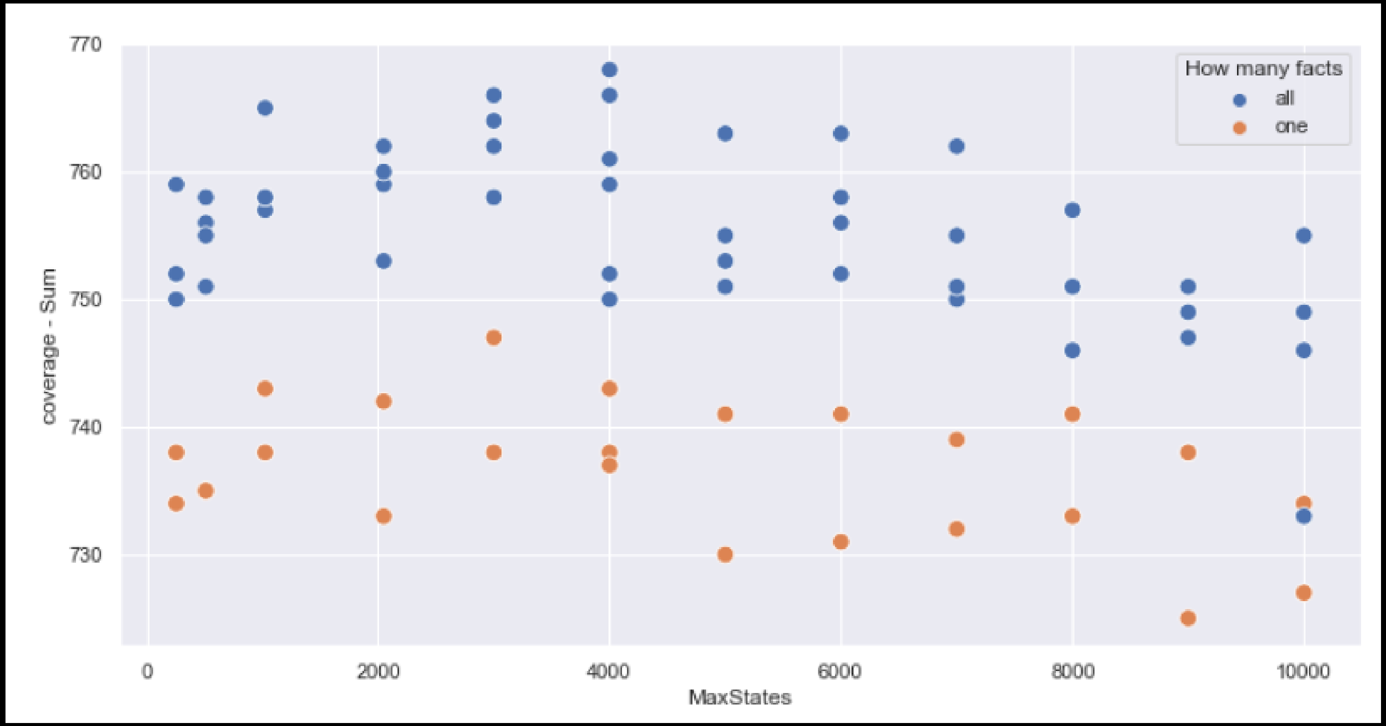
- One: Max refined domain is best
- In General splitting all facts is superior



How many FactPairs to split

- Same picture for coverage

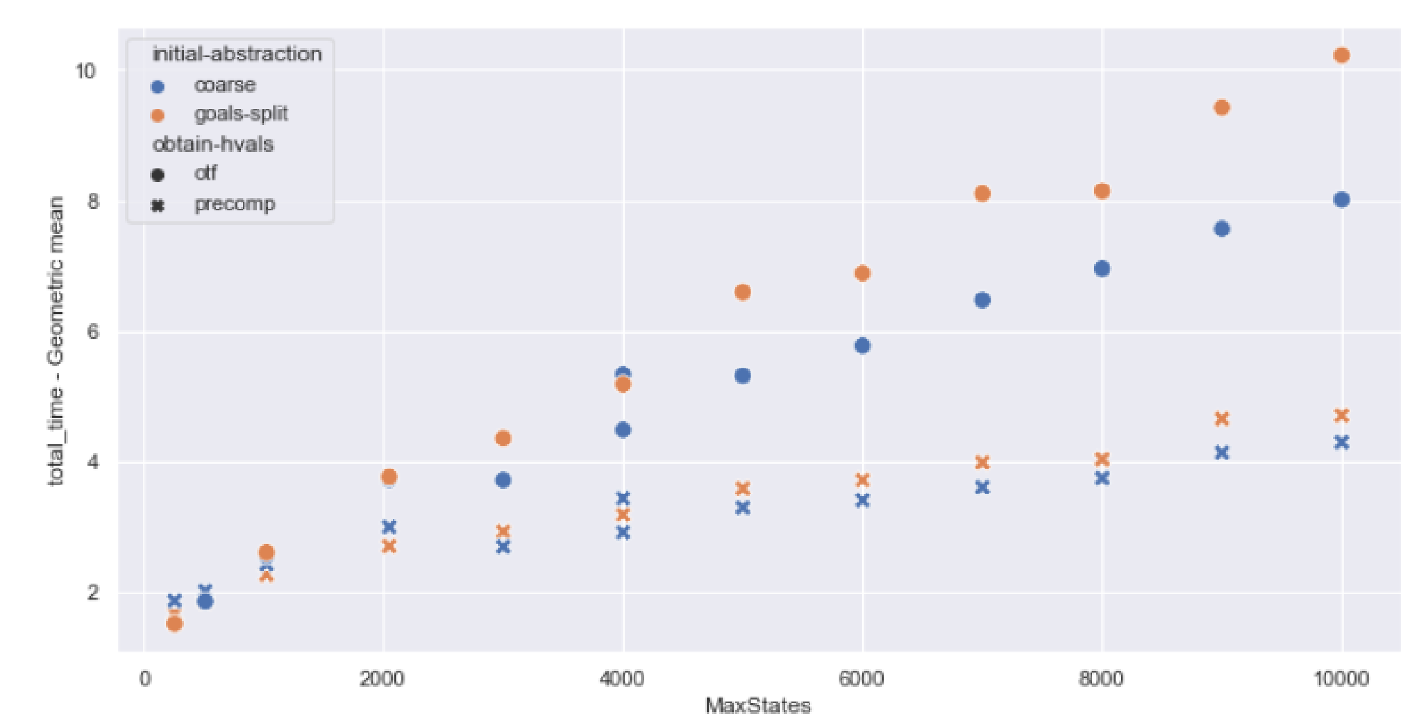
➡ Better to split all facts of a flaw (beneficial + faster refinement)



Initial abstraction

- Up to a 2000 statelimit initial goal split better
- Else most coarse abstraction superior

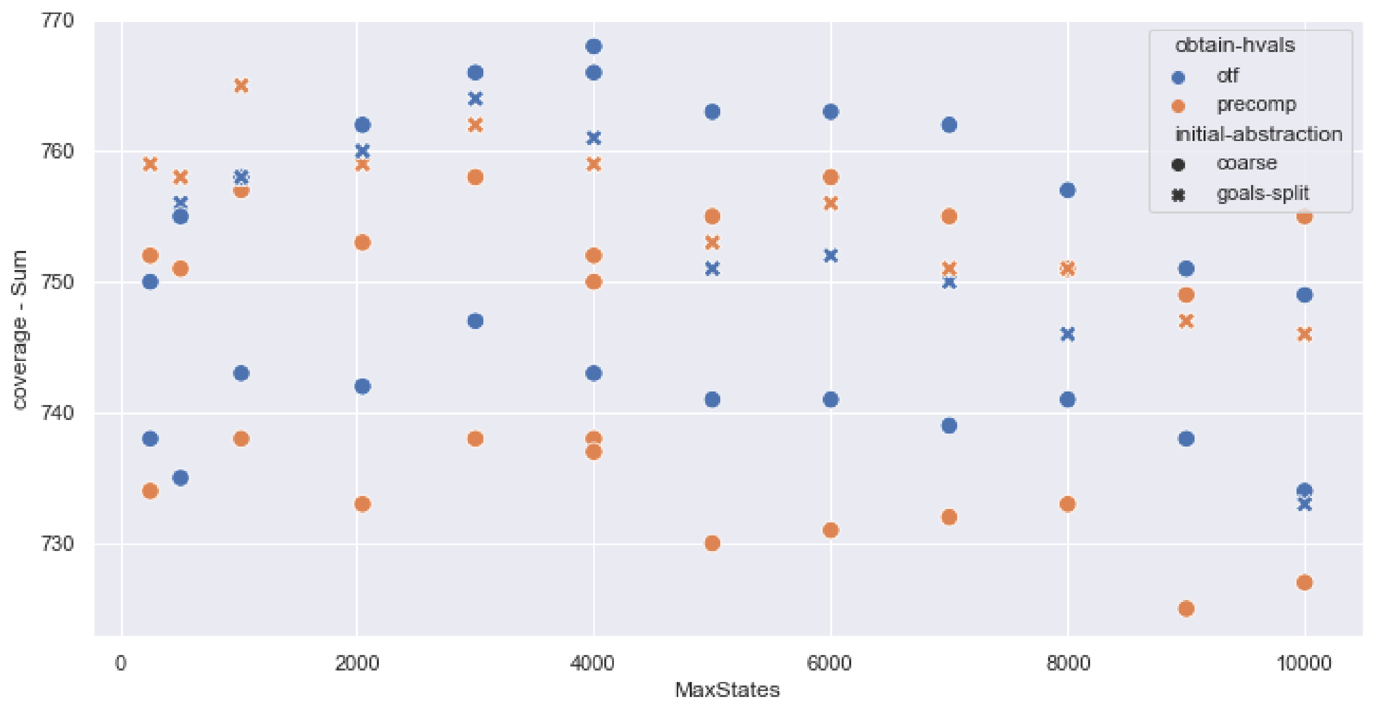
➡ Initial goal split good idea when less refinement opportunities



Obtain Heuristic values

- Mostly depends on statelimit
- Up to 2000 States and after 16000 States precomputation is superior
- Else „On demand“ yields best performing configurations

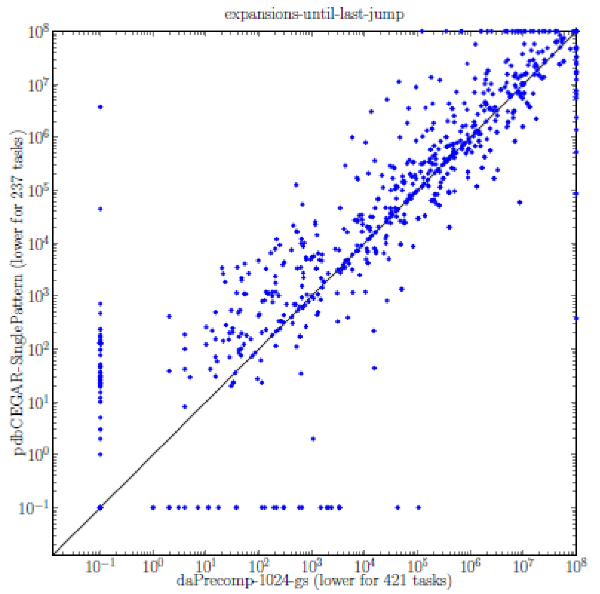
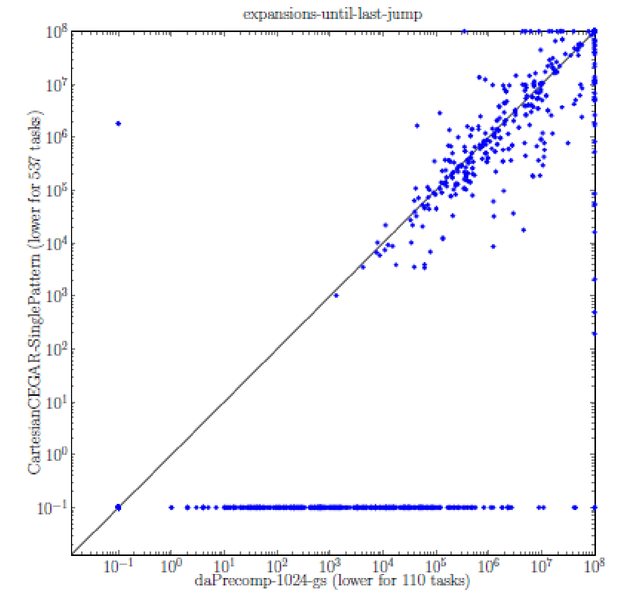
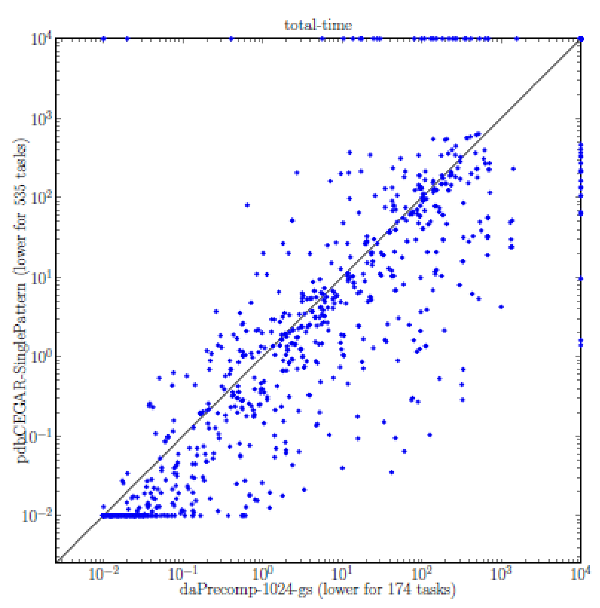
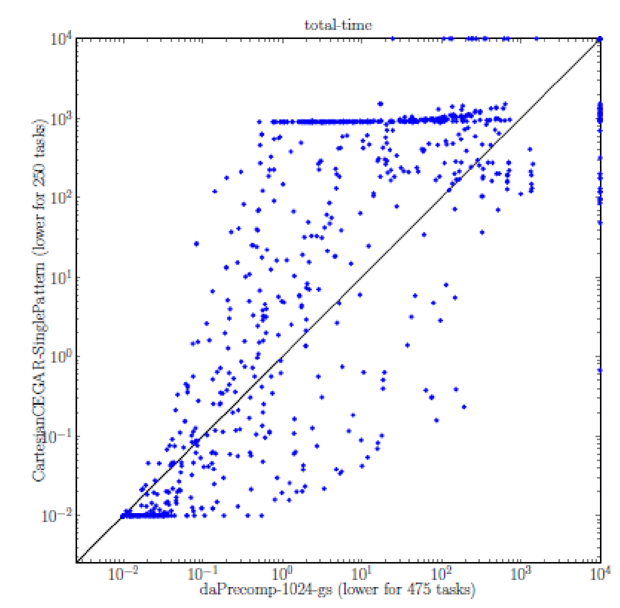
➡ Tradeoff between search time and time needed for backward search



Comparison

Algorithm:	$\boldsymbol{PDB^{SA}}$	$\boldsymbol{DA^{OTF}}$	$\boldsymbol{DA^{Precomp}}$	$\boldsymbol{Cartesian^{SA}}$
Coverage:	761	765	764	**791**

- $\boldsymbol{PDB^{SA}}$: constructs one single pattern using the cegar principle
- $\boldsymbol{DA^{OTF}}$: sizelimit 4000, obtain h-vals on demand, no initial goal split
- $\boldsymbol{DA^{Precomp}}$: sizelimit 1024, precomputation, initial goal split
- $\boldsymbol{Cartesian^{SA}}$: constructs one single cartesian abstraction using the cegar principle


expansions-until-last-jump
pdbCEGAR-SinglePattern (lower for 237 tasks)
daPrecomp-1024-gs (lower for 421 tasks)

expansions-until-last-jump
CartesianCEGAR-SinglePattern (lower for 537 tasks)
daPrecomp-1024-gs (lower for 110 tasks)

total-time
pdbCEGAR-SinglePattern (lower for 535 tasks)
daPrecomp-1024-gs (lower for 174 tasks)

total-time
CartesianCEGAR-SinglePattern (lower for 250 tasks)
daPrecomp-1024-gs (lower for 475 tasks)

Comparison with Multi-Abstraction Methods

Algorithm:	PDB^{SA}	DA^{OTF}	$DA^{Precomp}$	$Cartesian^{SA}$
Coverage:	761	765	764	**791**

Algorithm:	PDB^{add}	PDB^{nadd}	$Cartesian^{MA}$
Coverage:	862	900	889

- PDB^{nadd}, PDB^{add}: Use cegar principle to construct multiple Projections
- $Cartesian^{MA}$: constructs multiple cartesian abstractions

→ **Significant performance-gain compared to single abstraction methods**

Conclusion

And Future Work

Conclusion

- Developed and Implemented a capable Algorithm for the construction of Domain Abstractions.
- Performance ranks in between CEGAR-Algorithms for Projections and Cartesian Abstractions (Single Abstraction)

Next Steps

- Extend Algorithm for the construction of multiple Abstractions
- Split n FactPairs / Goals
- Regroup Values in domains (Simulated annealing)
- Comprehensive experiments to compare all possible parameter combinations