University of Basel

Modelling Git Operations as Planning Problems

Tim Bachmann <tim.bachmann@stud.unibas.ch>

Department Mathematics and Computer Science, University of Basel

3. February 2021

Motivation

- Modelling graph operations as planning tasks
- Creating a new benchmarks for planners

Planning

Planning Problem:

- Set of variables.
- State: Values assigned to all variables.
- Actions: Relation between two states.

Goal:

- Finding a sequence of actions from **initial state** to **goal state**.

Problem Domain Definition Language

Domain File:

- Predicates
- Derived Predicates
- Actions

Actions:

- Parameters
- Preconditions
- Effect

Problem File:

- Objects
- Initial State
- Goal States

Problem Domain Definition Language

Domain File:

- Predicates
- Derived Predicates
- Actions

Actions:

- Parameters
- Preconditions
- Effect

Problem File:

- Objects
- Initial State
- Goal States

Problem Domain Definition Language

Domain File:

- Predicates
- Derived Predicates
- Actions

Actions:

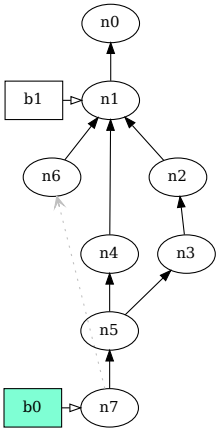
- Parameters
- Preconditions
- Effect

Problem File:

- Objects
- Initial State
- Goal States

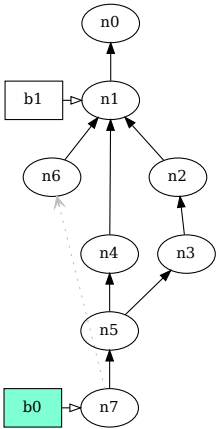
Git State

- Nodes (Commits)
- Branch Pointers
- HEAD Pointer
- Copy Relation

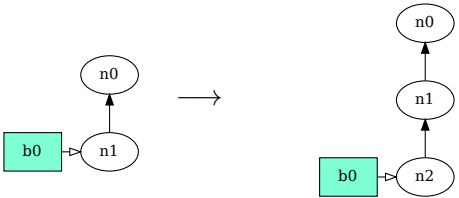


Git State

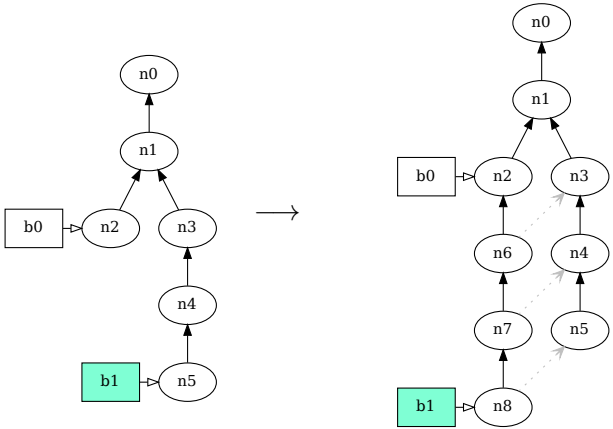
- Nodes (Commits)
- Branch Pointers
- HEAD Pointer
- Copy Relation



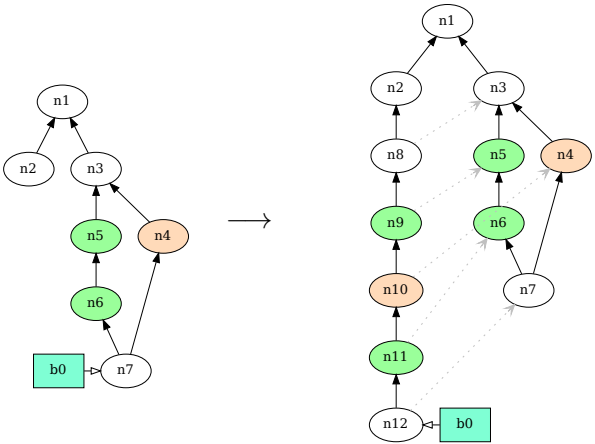
Git Commit



Git Rebase - Simple Example



Git Rebase - Advanced Example



The Git State

Objects:

- Nodes
- Branches

Predicates:

- `(n_is_parent ?n1 ?n2)`
- `(b_points_to ?b ?n)`
- `(is_in_graph ?o)`
- `(is_head ?bn)`
- `(n_is_copy_of ?n1 ?n2)`

Derived Predicates:

- `(n_is_ancestor ?n1 ?n2)`

The Git State

Objects:

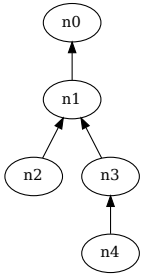
- Nodes
- Branches

Predicates:

- `(n_is_parent ?n1 ?n2)`
- `(b_points_to ?b ?n)`
- `(is_in_graph ?o)`
- `(is_head ?bn)`
- `(n_is_copy_of ?n1 ?n2)`

Derived Predicates:

- `(n_is_ancestor ?n1 ?n2)`



The Git State

Objects:

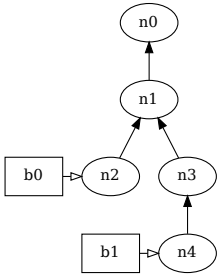
- Nodes
- Branches

Predicates:

- `(n_is_parent ?n1 ?n2)`
- `(b_points_to ?b ?n)`
- `(is_in_graph ?o)`
- `(is_head ?bn)`
- `(n_is_copy_of ?n1 ?n2)`

Derived Predicates:

- `(n_is_ancestor ?n1 ?n2)`



The Git State

Objects:

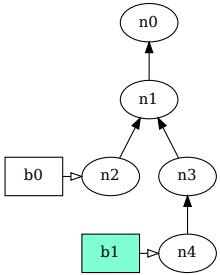
- Nodes
- Branches

Predicates:

- `(n_is_parent ?n1 ?n2)`
- `(b_points_to ?b ?n)`
- `(is_in_graph ?o)`
- `(is_head ?bn)`
- `(n_is_copy_of ?n1 ?n2)`

Derived Predicates:

- `(n_is_ancestor ?n1 ?n2)`



The Git State

Objects:

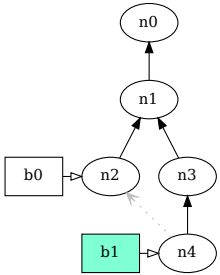
- Nodes
- Branches

Predicates:

- `(n_is_parent ?n1 ?n2)`
- `(b_points_to ?b ?n)`
- `(is_in_graph ?o)`
- `(is_head ?bn)`
- `(n_is_copy_of ?n1 ?n2)`

Derived Predicates:

- `(n_is_ancestor ?n1 ?n2)`



The Git State

Objects:

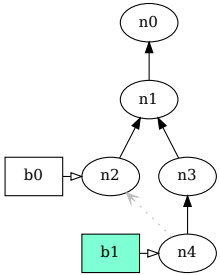
- Nodes
- Branches

Predicates:

- `(n_is_parent ?n1 ?n2)`
- `(b_points_to ?b ?n)`
- `(is_in_graph ?o)`
- `(is_head ?bn)`
- `(n_is_copy_of ?n1 ?n2)`

Derived Predicates:

- `(n_is_ancestor ?n1 ?n2)`



Actions - Git Commit

Parameters: HEAD node, HEAD branch, new node.

Preconditions:

```
(and
    (not (is_in_graph ?new_node))
    (is_head ?head_branch)
    (b_points_to ?head_branch ?head_node )
)
```

Effect:

```
(and
    (is_in_graph ?new_node)
    (n_is_parent ?head_node ?new_node)
    (b_points_to ?head_branch ?new_node )
    (not (b_points_to ?head_branch ?head_node ))
)
```

Actions - Git Commit

Parameters: HEAD node, HEAD branch, new node.

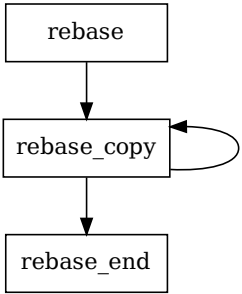
Preconditions:

```
(and
    (not (is_in_graph ?new_node))
    (is_head ?head_branch)
    (b_points_to ?head_branch ?head_node )
)
```

Effect:

```
(and
    (is_in_graph ?new_node)
    (n_is_parent ?head_node ?new_node)
    (b_points_to ?head_branch ?new_node )
    (not (b_points_to ?head_branch ?head_node ))
)
```

Actions - Git Rebase



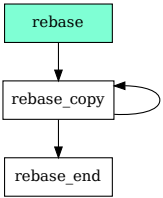
Actions - Git Rebase

Parameters: HEAD node, Target node, LCA node
Preconditions:

```
(and
    (n_is_head ?head)
    (n_is_lowest_common_ancestor ?LCA ?head ?target)
    (not (n_is_ancestor ?head ?target))
    (not (n_is_ancestor ?target ?head))
    (not (mode_rebase))
)
```

Effects:

```
(and
    (mode_rebase)
    (rebase_state ?head ?target)
    (forall (?n - node) (when (and
        (n_is_parent ?LCA ?n)
        (or (n_is_ancestor ?n ?head)
            (= ?n ?head)))
        (rebase_to_copy ?n)))
)
```



Actions - Git Rebase

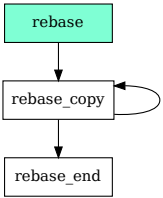
Parameters: HEAD node, Target node, LCA node

Preconditions:

```
(and
    (n_is_head ?head)
    (n_is_lowest_common_ancestor ?LCA ?head ?target)
    (not (n_is_ancestor ?head ?target))
    (not (n_is_ancestor ?target ?head))
    (not (mode_rebase))
)
```

Effects:

```
(and
    (mode_rebase)
    (rebase_state ?head ?target)
    (forall (?n - node) (when (and
        (n_is_parent ?LCA ?n)
        (or (n_is_ancestor ?n ?head)
            (= ?n ?head)))
        (rebase_to_copy ?n)))
)
```



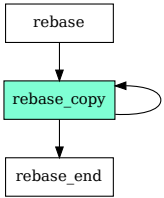
Actions - Git Rebase - copy

Parameters:

HEAD node, Target node, Copy node, New node

Effects:

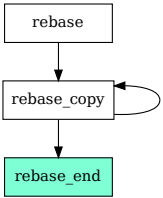
```
(and
    (is_in_graph ?new)
    (n_is_parent ?target ?new)
    (n_is_copy_of ?new ?copy)
    (rebase_is_copied ?copy)
   ;( ... )
    (not (rebase_to_copy ?copy))
    (not (rebase_state ?head ?target))
    (rebase_state ?head ?new))
)
```



Actions - Git Rebase - end

Preconditions:

```
(and
    ;(...)
     (not (exists (?n - node)
          (rebase_to_copy ?n)))
)
```



Other Domain Model: Without Derived Predicates

Motivation:

- Reducing the number of axioms
- Finding performance differences

- Manually compute the predicate (`n_is_ancestor`)
- Replace derived predicates with quantifiers (`n_is_lowest_common_ancestor`)

Other Domain Model: Without Derived Predicates

Motivation:

- Reducing the number of axioms
- Finding performance differences

- Manually compute the predicate (`n_is_ancestor`)
- Replace derived predicates with quantifiers (`n_is_lowest_common_ancestor`)

Other Domain Model: Ordered Nodes and Branches

Motivation:

- Reduce the size of the state space.
- Improve the performance of the solver.

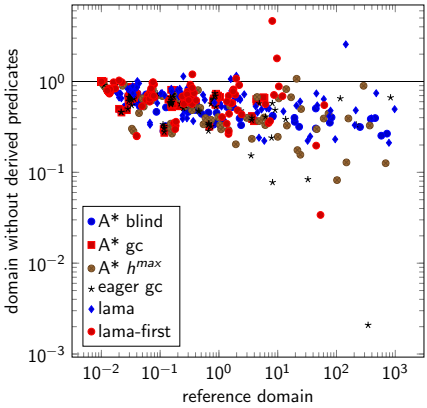
Method:

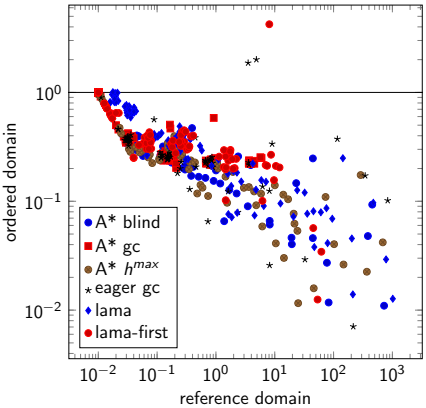
- `(current_object ?o)`
- `(next_object ?o1 ?o2)`

Benchmarks

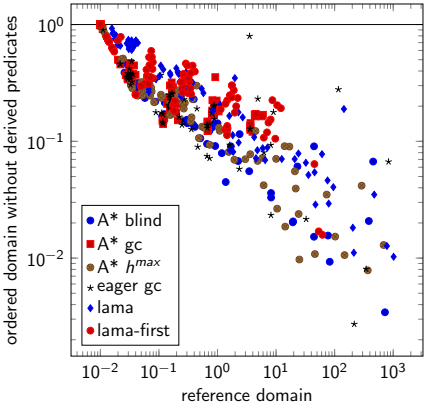
- Generate equivalent problems for all domains
- Multiple complexities: 1 to 15 nodes and branches.
- Let planners solve each problem
- Total: **138 Problem Files**

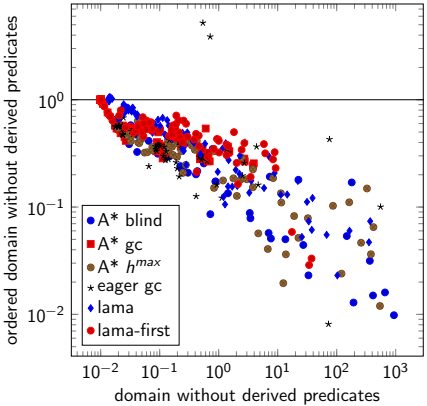
Results: Time



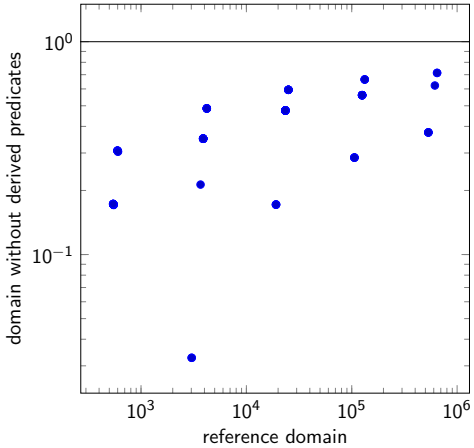


Results: Time





Results: Axioms



Conclusion

- Modeling Git operation in PDDL
- Considerable search time differences between domain variants.
- Reducing the amount of axioms is possible

Summary

- Git graph as predicates
- Git commands as actions
- Multiple domain variants
  - Reference domain
  - Without derived predicates
  - With ordered PDDL objects
  - With both modifications
- Compared planners with the domains

PDDL Git Commit

```
(:action commit :parameters (?n_h ?n_star - node ?h - branch)
    :precondition (and
        (not (is_in_graph ?n_star))
        (is_head ?h)
        (b_points_to ?h ?n_h )
        (not (mode_rebase)))
    :effect (and
        (is_in_graph ?n_star)
        (n_is_parent ?n_h ?n_star)
        (b_points_to ?h ?n_star )
        (not (b_points_to ?h ?n_h ))
        (increase (total-cost) 1)))
```

PDDL Git Rebase

```
(:action rebase :parameters (?head ?target ?LCA - node)
    :precondition (and
        (n_is_head ?head)
        (n_is_lowest_common_ancestor ?LCA ?head ?target)
        (not (n_is_ancestor ?head ?target))
        (not (n_is_ancestor ?target ?head))
        (not(mode_rebase)))
    :effect (and
        (mode_rebase)
        (rebase_state ?head ?target)
        (forall (?n - node) (when (and
            (n_is_parent ?LCA ?n)
            (or (n_is_ancestor ?n ?head)
                (= ?n ?head)))
            (rebase_to_copy ?n)))
        (increase (total-cost) 1)))
```

PDDL Git Rebase Copy

```
(:action rebase__copy :parameters (?head ?target ?copy ?new - node)
    :precondition (and
        (mode_rebase)
        (rebase_state ?head ?target)
        (rebase_to_copy ?copy)
        (not (is_in_graph ?new)))
    :effect (and
        (is_in_graph ?new)
        (n_is_parent ?target ?new)
        (n_is_copy_of ?new ?copy)
        (rebase_is_copied ?copy)
        (forall (?n - node) (when (and
            (n_is_parent ?copy ?n)
            (not (rebase_is_copied ?n))
            (or (n_is_ancestor ?n ?head)
                (= ?n ?head)))
            (rebase_to_copy ?n)))
        (not (rebase_to_copy ?copy))
        (not (rebase_state ?head ?target))
        (rebase_state ?head ?new)))
```

PDDL Git Rebase End

```
(:action rebase__end :parameters (?n_h ?n_lc - node ?h - branch)
    :precondition (and
        (mode_rebase)
        (rebase_state ?n_h ?n_lc)
        (is_head ?h)
        (not (exists (?n - node)
          (rebase_to_copy ?n))))
    :effect (and
        (not (mode_rebase))
        (not (rebase_state ?n_h ?n_lc))
        (forall (?n - node) (not (rebase_is_copied ?n)))
        (not (b_points_to ?h ?n_h))
        (b_points_to ?h ?n_lc)))
```

PDDL Lowest Common Ancestor

```
(:derived
  (n_is_first_common_ancestor ?n_common ?n1 ?n2 - node)
  (and
    (n_is_ancestor ?n_common ?n1)
    (n_is_ancestor ?n_common ?n2)
    (not (= ?n1 ?n2))
    (not (exists
        (?n_child - node)
        (and
          (n_is_parent ?n_common ?n_child)
          (n_is_ancestor ?n_child ?n1)
          (n_is_ancestor ?n_child ?n2))))
  )
)
```