Incremental and Non-incremental Learning of Control Knowledge for Planning

Daniel Borrajo Millán
joint work with Manuela Veloso, Ricardo Aler, and Susana Fernández

Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Madrid, SPAIN
Web: http://scalab.uc3m.es/∼dborrajo

Incremental and Non-incremental Learning of Control Knowledge for Planning

1. **Motivation**
2. Incremental learning. HAMLET
3. Learning by genetic programming. EVOCK
4. Discussion

Motivation for HAMLET

- Control knowledge learning techniques that worked well for linear planning, had problems in nonlinear planning

Motivation for HAMLET

- Control knowledge learning techniques that worked well for linear planning, had problems in nonlinear planning
- EBL
  - generated over-general or over-specific control knowledge
  - sometimes they required domain axioms
  - utility and expensive chunk problems

Motivation for HAMLET

- Control knowledge learning techniques that worked well for linear planning, had problems in nonlinear planning
- EBL
  - generated over-general or over-specific control knowledge
  - sometimes they required domain axioms
  - utility and expensive chunk problems
- Pure inductive techniques
  - did not use available domain knowledge: difficulty to focus on what is important
  - required powerful representation mechanisms beyond attribute-value: predicate logic (ILP)
  - huge hypothesis spaces very difficult to search without the use of learning heuristics

Our solution

Incremental approach

- Learning task:
  - Given: a domain theory, a set of training problems (it might be empty), a set of initial control rules (usually empty), and a set of parameters (quality metric, learning time bound, modes, . . .)
  - Output: a set of control rules that "efficiently" solves test problems generating "good quality" solutions

Our solution

Incremental approach

- Learning task:
  - Given: a domain theory, a set of training problems (it might be empty), a set of initial control rules (usually empty), and a set of parameters (quality metric, learning time bound, modes, . . .)
  - Output: a set of control rules that "efficiently" solves test problems generating "good quality" solutions
- Main idea:
  - Uses EBL for acquiring control rules from problem solving traces
  - Uses relational induction (in the spirit of version spaces) to generalize and specialize control rules

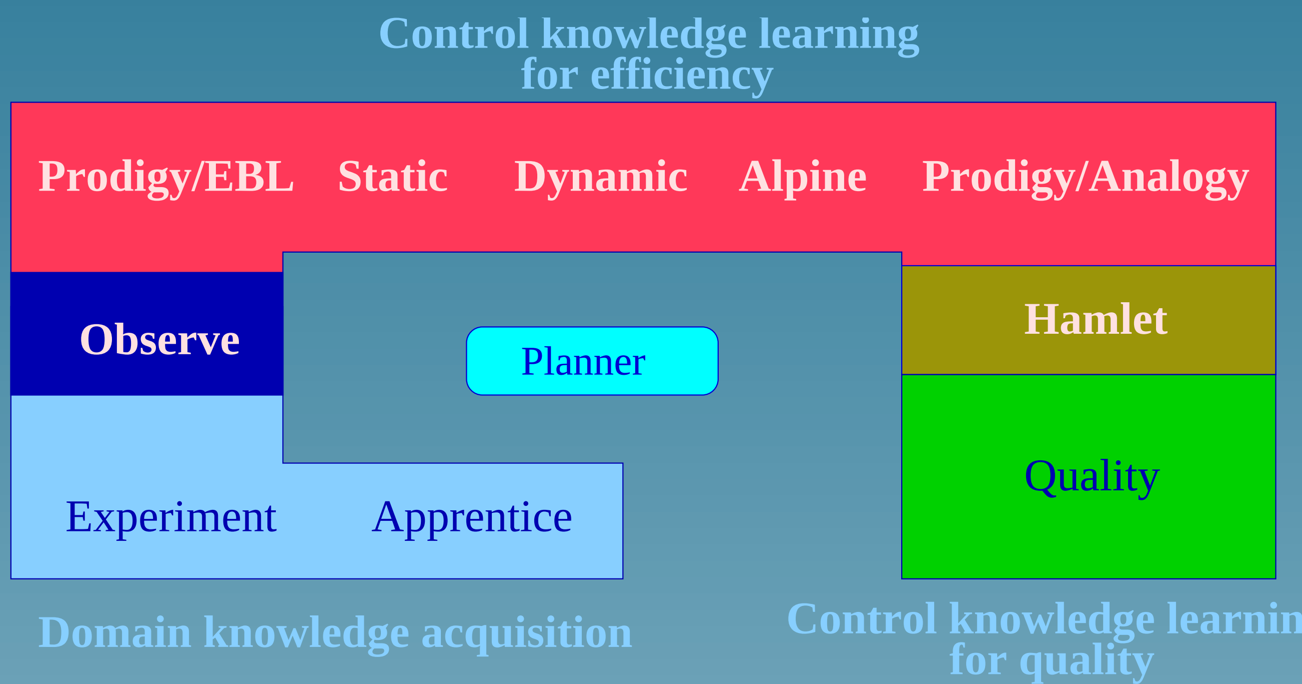
Incremental and Non-incremental Learning of Control Knowledge for Planning

1. Motivation
2. **Incremental learning.** HAMLET
3. Learning by genetic programming. EVOCK
4. Discussion

Hybrid Learning. HAMLET

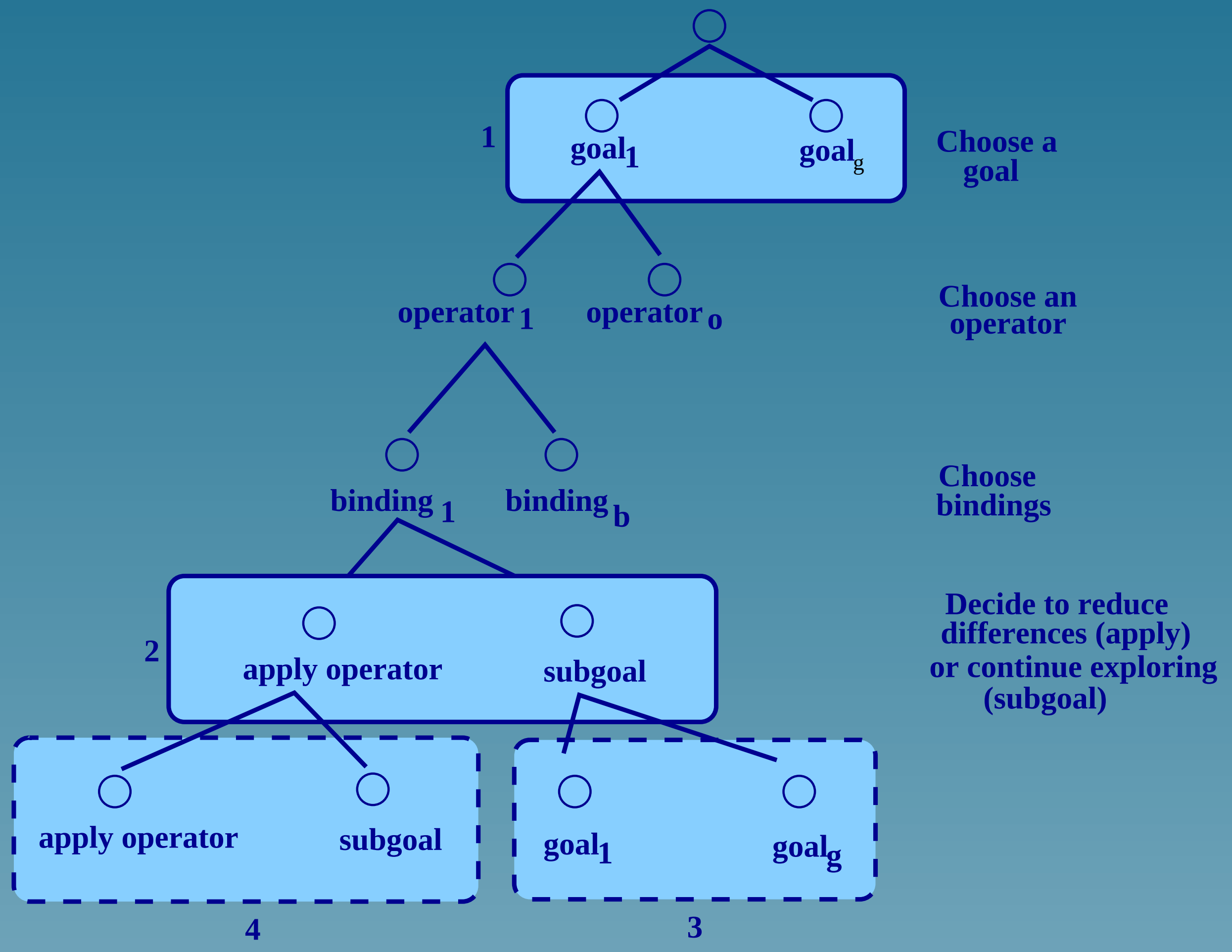
Planning architecture. PRODIGY

- Integrated architecture for non-linear problem solving and learning
- Means-ends analysis with bidirectional search

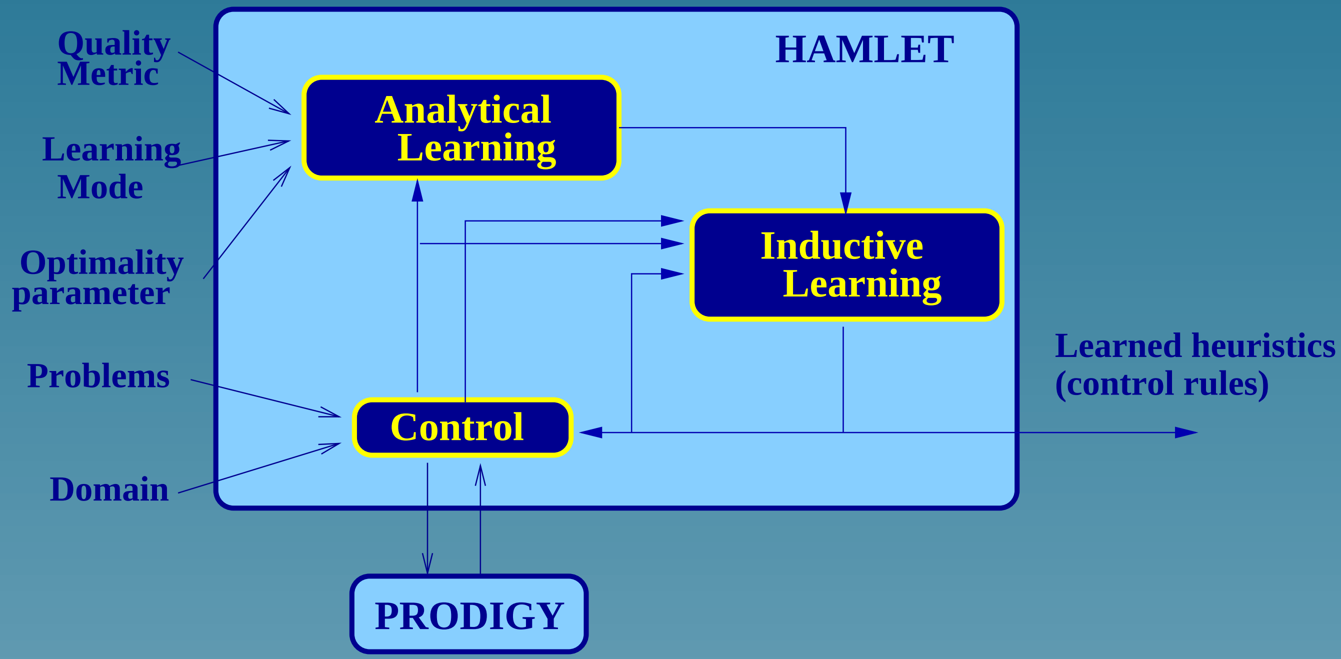


Hybrid Learning. HAMLET

PRODIGY search tree



Incremental learning. HAMLET



Example of control rule

```
(control-rule select-operators-unload-airplane
    (if (current-goal (at <object> <location1>))
        (true-in-state (at <object> <location2>))
        (true-in-state (loc-at <location1> <city1>))
        (true-in-state (loc-at <location2> <city2>))
        (type-of-object <object> object)
        (type-of-object <location1> location))
    (then select operator unload-airplane))
```

Example of control rule

```
(control-rule select-operators-unload-airplane
    (if (current-goal (at <object> <location1>))
        (true-in-state (at <object> <location2>))
        (true-in-state (loc-at <location1> <city1>))
        (true-in-state (loc-at <location2> <city2>))
        (type-of-object <object> object)
        (type-of-object <location1> location))
    (then select operator unload-airplane))
```

Difficulties:

- variables have to be bound to different values (cities)
- constants have to be of a specific type (`object` and `location1`)
- there are conditions that might not relate to the goal regression (`loc-at`)

Target concepts representation

(control-rule *name*
  (if (current-goal *goal-name*)
      [(prior-goals (*literal**))]
      (true-in-state *literal*)*
      (other-goals (*literal**))
      (type-of-object *object type*)*)
  (then select **operators** *operator-name*))

(control-rule *name*
  (if (and (applicable-op *operator*)
      [(prior-goals (*literal**))]
      (true-in-state *literal*)*
      (other-goals (*literal**))
      (type-of-object *object type*)*))
  (then decide {**apply**|**sub-goal**}))

(control-rule *name*
  (if (and (current-operator *operator-name*)
      (current-goal *goal-name*)
      [(prior-goals (*literal**))]
      (true-in-state *literal*)*
      (other-goals (*literal**))
      (type-of-object *object type*)*))
  (then select **bindings** *bindings*))

(control-rule *name*
  (if (and (target-goal *literal*)
      [(prior-goals (*literal**))]
      (true-in-state *literal*)*
      (other-goals (*literal**))
      (type-of-object *object type*)*))
  (then select **goals** *literal*))

Analytical learning

- The Bounded Explanation module (EBL)
  - **extracts positive examples** of the decisions made from the search trees
  - **generates control rules** from them selecting their preconditions

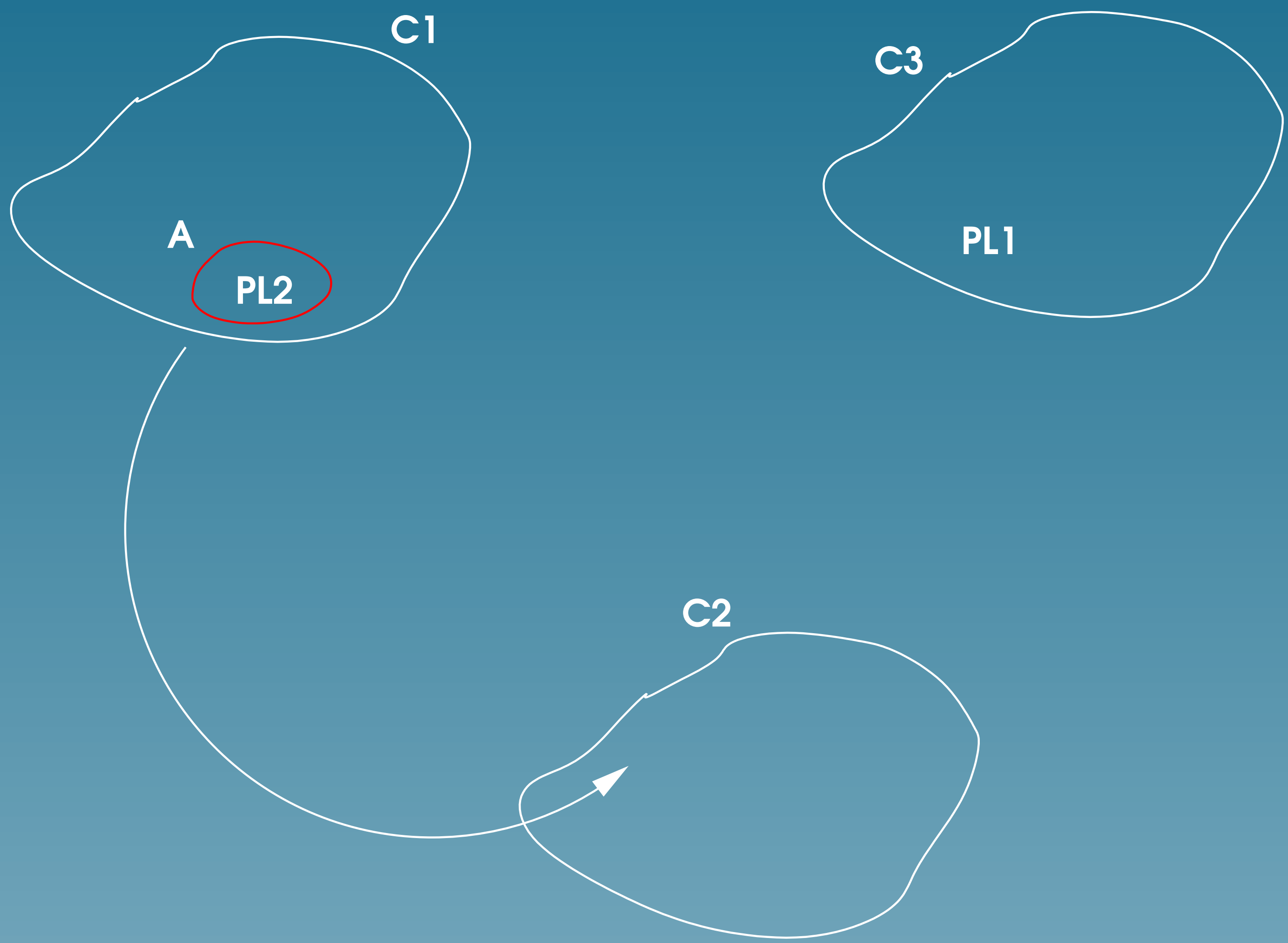
Analytical learning

- The Bounded Explanation module (EBL)
  - **extracts positive examples** of the decisions made from the search trees
  - **generates control rules** from them selecting their preconditions
- Target concepts:
  - **select** an unachieved **goal**
  - **select** an **operator** to achieve some **goal**
  - **select** bindings for an **operator** when trying to achieve a *goal*
  - **decide** to **apply an operator** for achieving a goal or **subgoal** on an unachieved **goal**

Analytical learning

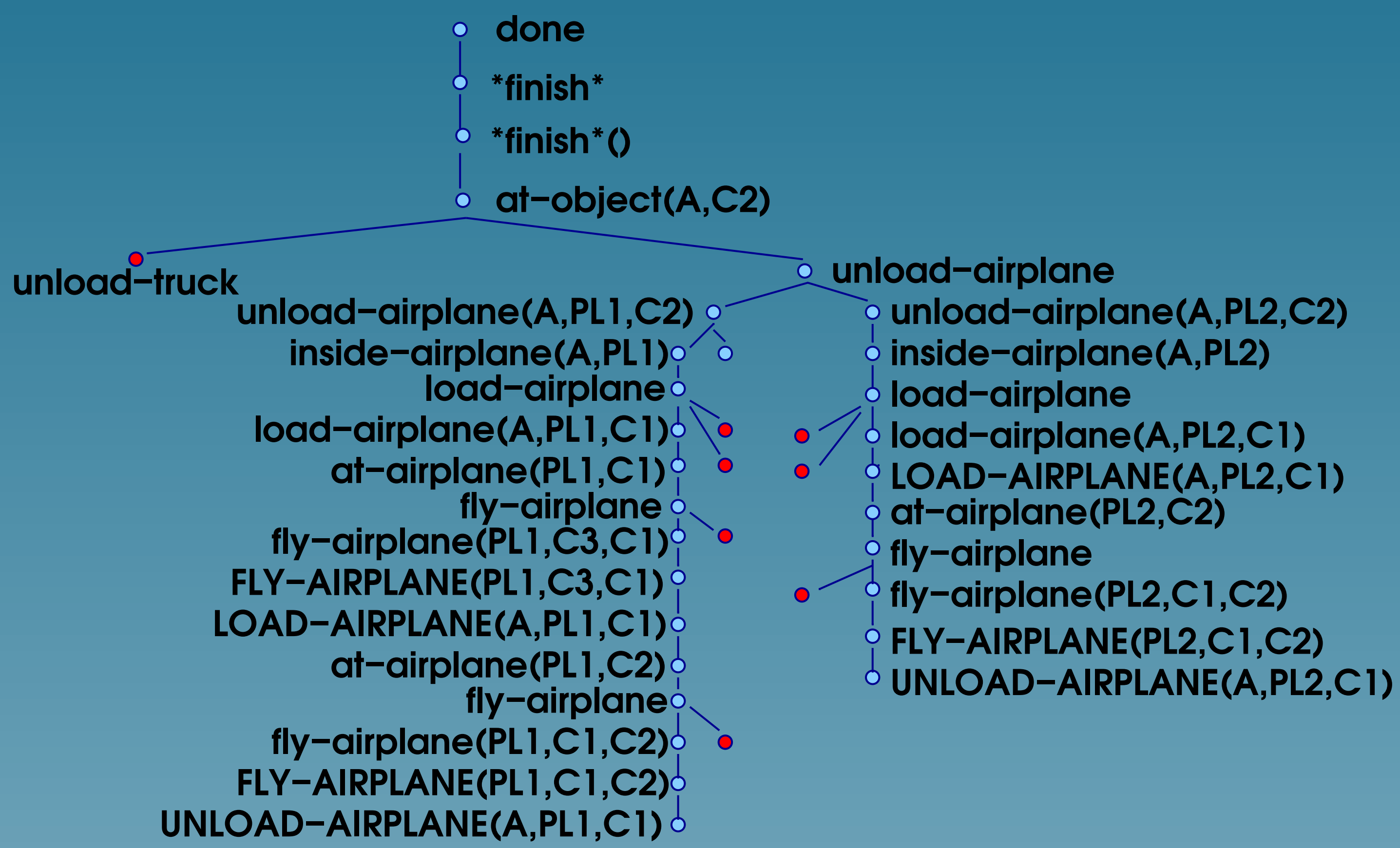
- The Bounded Explanation module (EBL)
  - **extracts positive examples** of the decisions made from the search trees
  - **generates control rules** from them selecting their preconditions
- Target concepts:
  - **select** an unachieved **goal**
  - **select** an **operator** to achieve some **goal**
  - **select** bindings for an **operator** when trying to achieve a *goal*
  - **decide** to **apply an operator** for achieving a goal or **subgoal** on an unachieved **goal**
- HAMLET considers multiple target concepts, each one being a disjunction of conjunctions (partially solves the utility problem)

Example of logistics problem



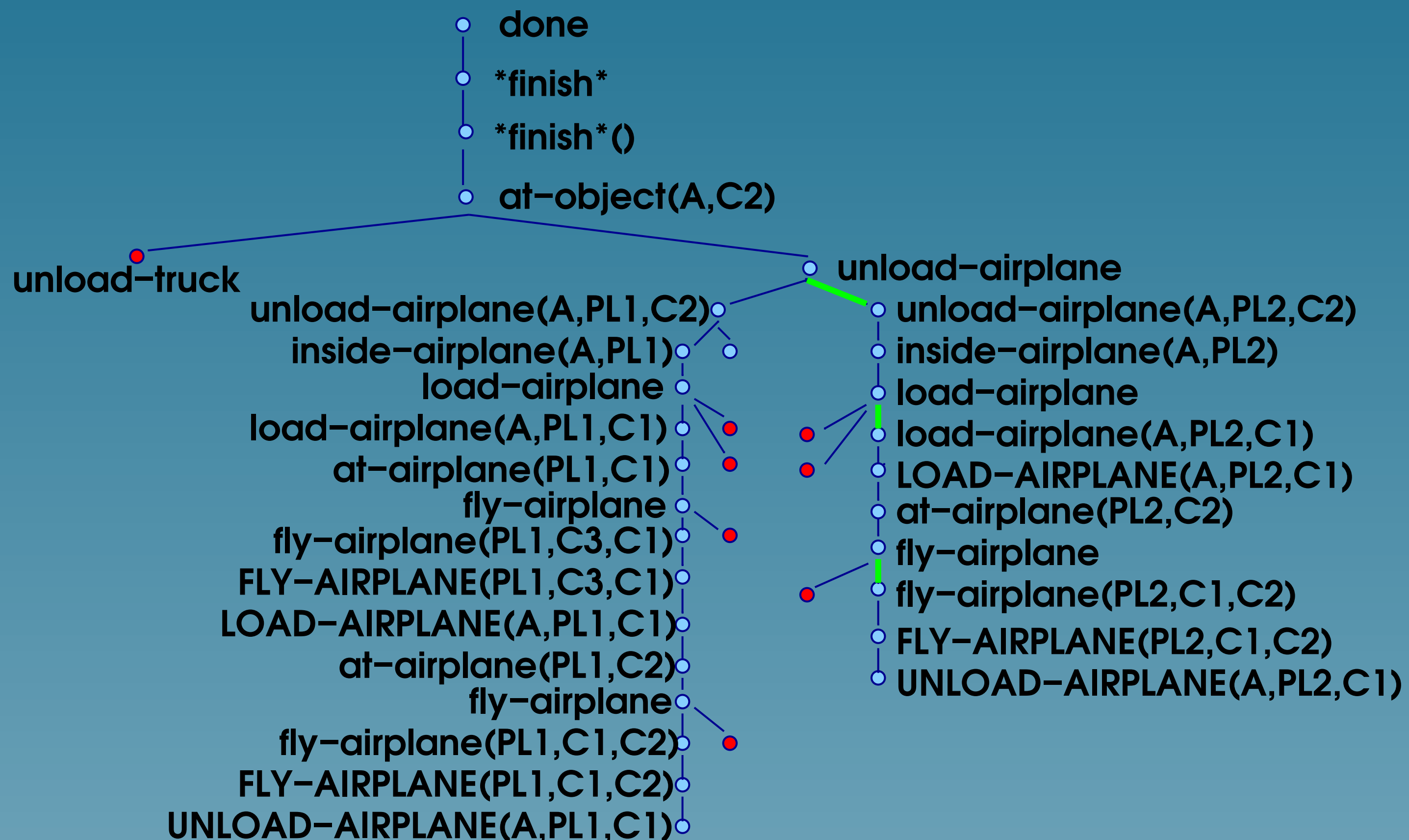
Hybrid Learning. HAMLET

Example of search tree



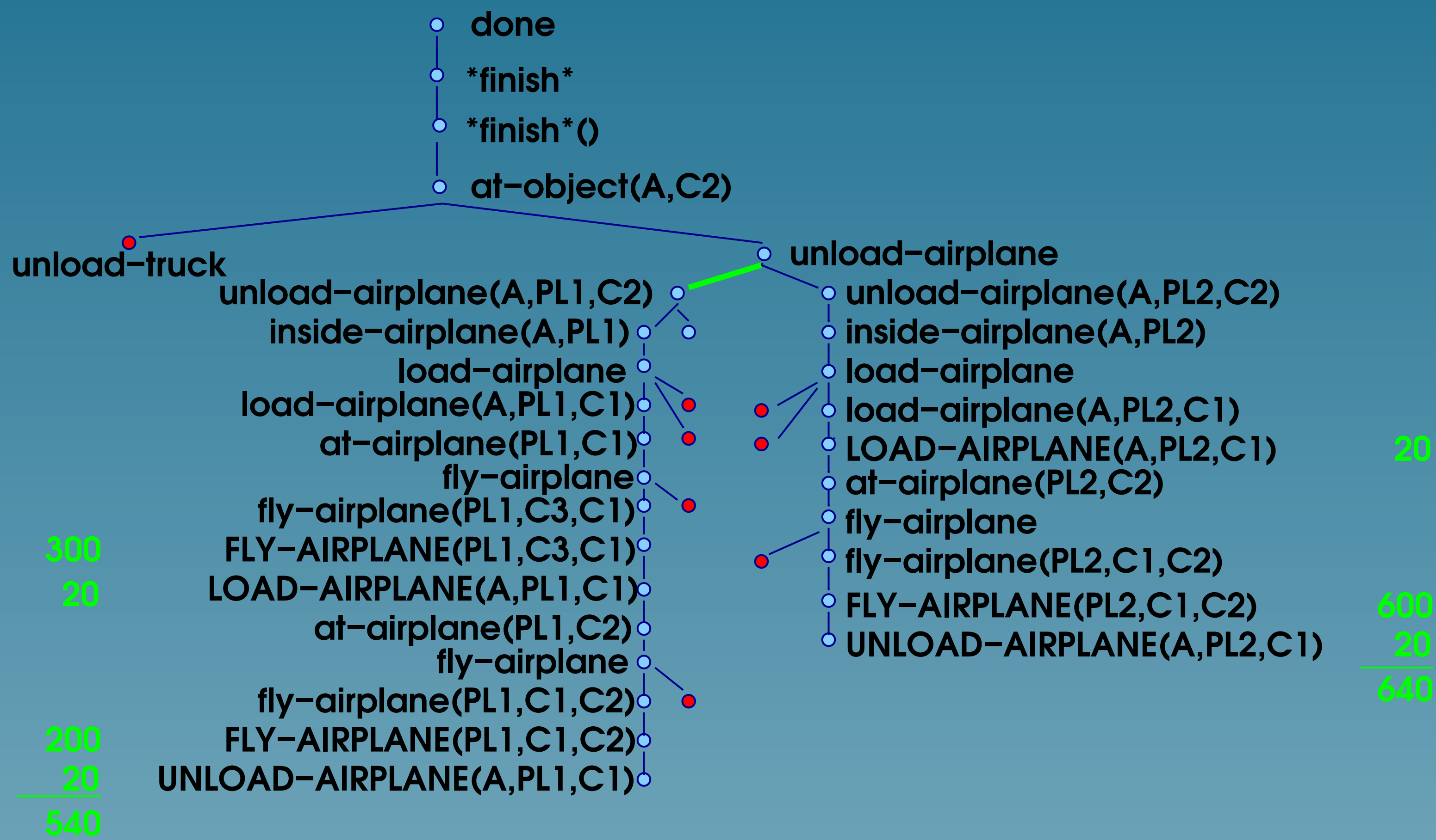
Hybrid Learning. HAMLET

Learning for plan length



Hybrid Learning. HAMLET

Learning for quality



Inductive learning. Generalization

```
(control-rule select-operators-unload-airplane
    (if (current-goal (at-object <object> <airport>))
        (true-in-state (inside-airplane <object> <plane>))
        (true-in-state (at-airplane <plane> <airport>)))
    (then select operator unload-airplane))
(control-rule select-operators-unload-airplane
    (if (current-goal (at-object <object> <airport>))
        (true-in-state (inside-airplane <object> <plane>))
        (true-in-state (at-airplane <plane> <airport1>)))
    (then select operator unload-airplane))
```

Hybrid Learning. HAMLET

Inductive learning. Generalization

(control-rule select-operators-unload-airplane
(if (current-goal (at-object <object> <airport>))
(true-in-state (inside-airplane <object> <plane>))
(true-in-state (at-airplane <plane> <airport>))))
(then select operator unload-airplane))
(control-rule select-operators-unload-airplane
(if (current-goal (at-object <object> <airport>))
(true-in-state (inside-airplane <object> <plane>))
(true-in-state (at-airplane <plane> <airport1>))))
(then select operator unload-airplane))
(control-rule select-operators-unload-airplane
(if (current-goal (at-object <object> <airport>))
(true-in-state (inside-airplane <object> <plane>)))
(then select operator unload-airplane))

Finding negative examples

- Negative example of a control rule: it was applied at some node that lead to a failure, or a worse solution than the best sibling solution
- Only the **most general** negative examples are stored for each target concept
- They serve two purposes
  - refine an overly general rule
  - establish an upper limit of generalization for future applications of the generalization operators

Inductive learning. Specialization

(control-rule select-operators-unload-airplane
    (if (current-goal (at-object <object> <airport>))
        (true-in-state (inside-airplane <object> <plane>))))
    (then select operator unload-airplane))

(control-rule select-operators-unload-airplane
    (if (current-goal (at-object <object> <airport>))
        (true-in-state (at-object <object> <airport1>))))
    (then select operator unload-airplane))

Inductive learning. Specialization

(control-rule select-operators-unload-airplane
(if (current-goal (at-object <object> <airport>))
(true-in-state (inside-airplane <object> <plane>))))
(then select operator unload-airplane))

(control-rule select-operators-unload-airplane
(if (current-goal (at-object <object> <airport>))
(true-in-state (at-object <object> <airport1>))))
(then select operator unload-airplane))

(control-rule select-operators-unload-airplane
(if (current-goal (at-object <object> <airport>)))
(then select operator unload-airplane))

Hybrid Learning. HAMLET

Incremental flavor

E1 +

E2 +

Hybrid Learning. HAMLET

Incremental flavor

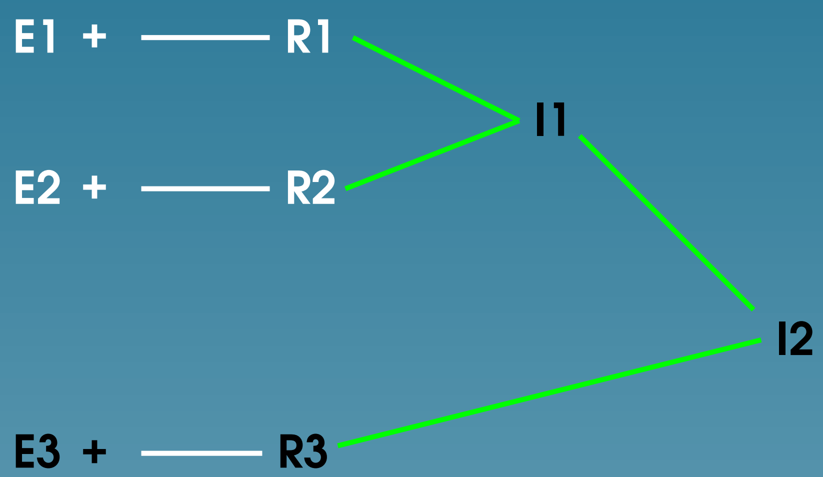
E1 + ——— R1

E2 + ——— R2

Hybrid Learning. HAMLET

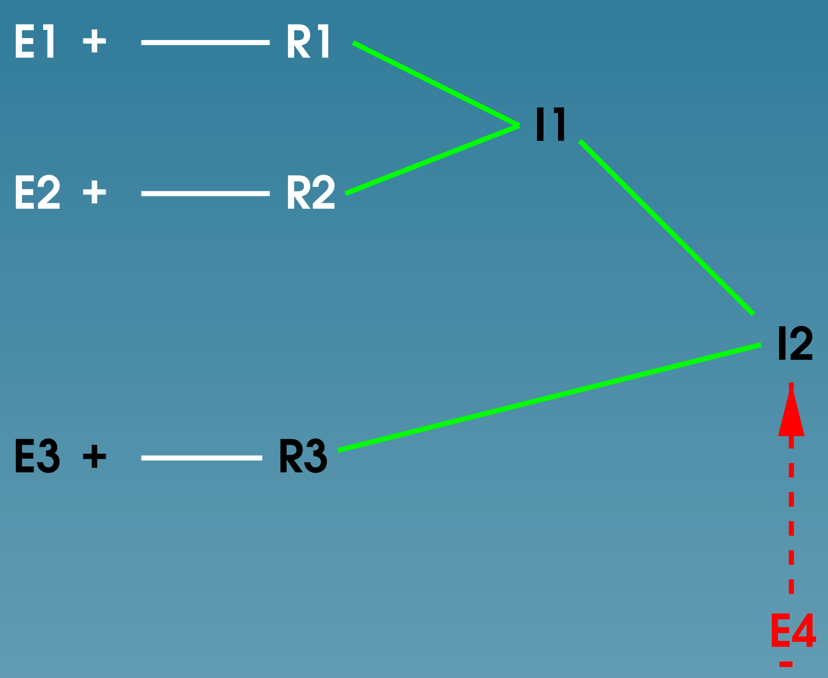
Incremental flavor

Incremental flavor



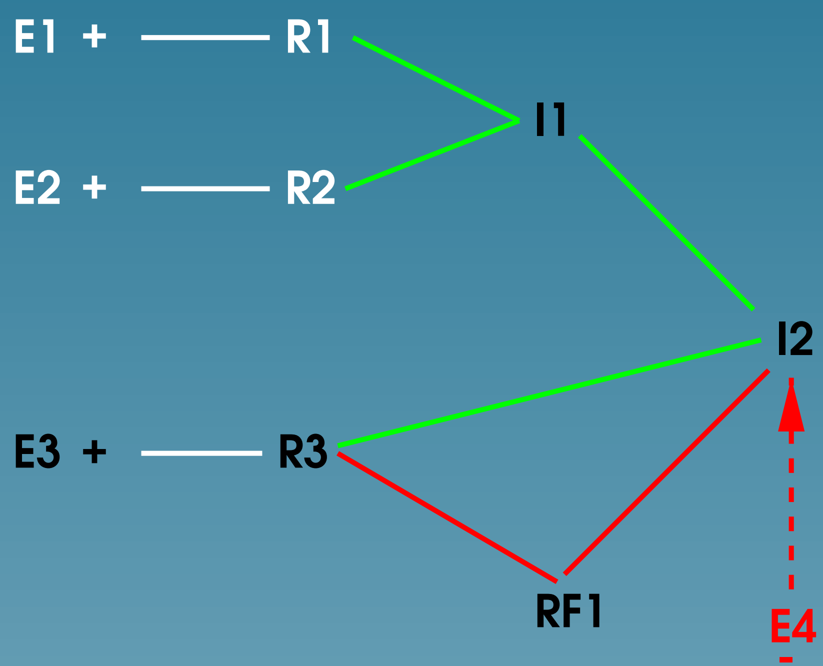
Hybrid Learning. HAMLET

Incremental flavor



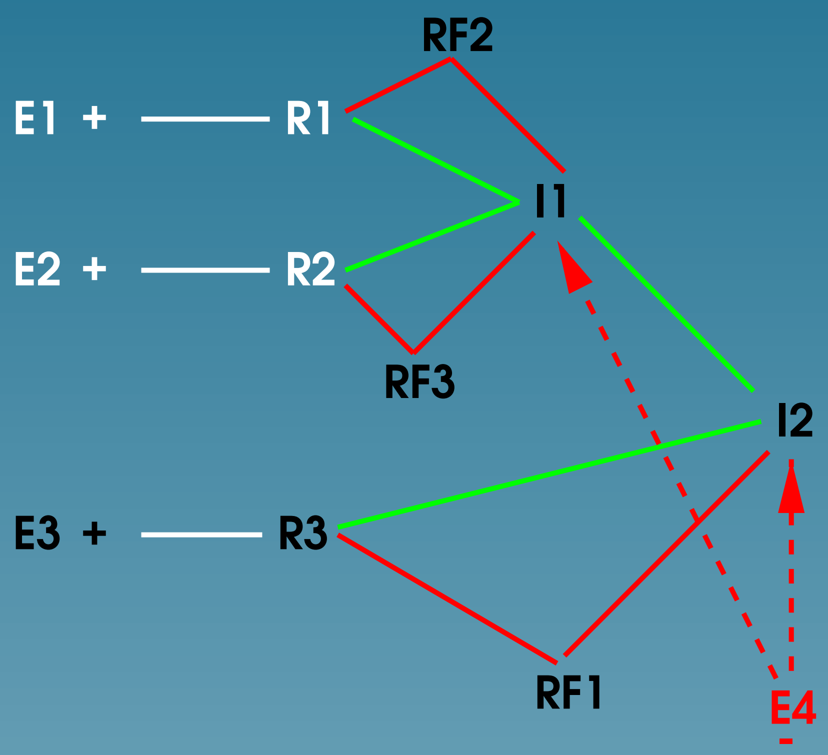
Hybrid Learning. HAMLET

Incremental flavor



Hybrid Learning. HAMLET

Incremental flavor



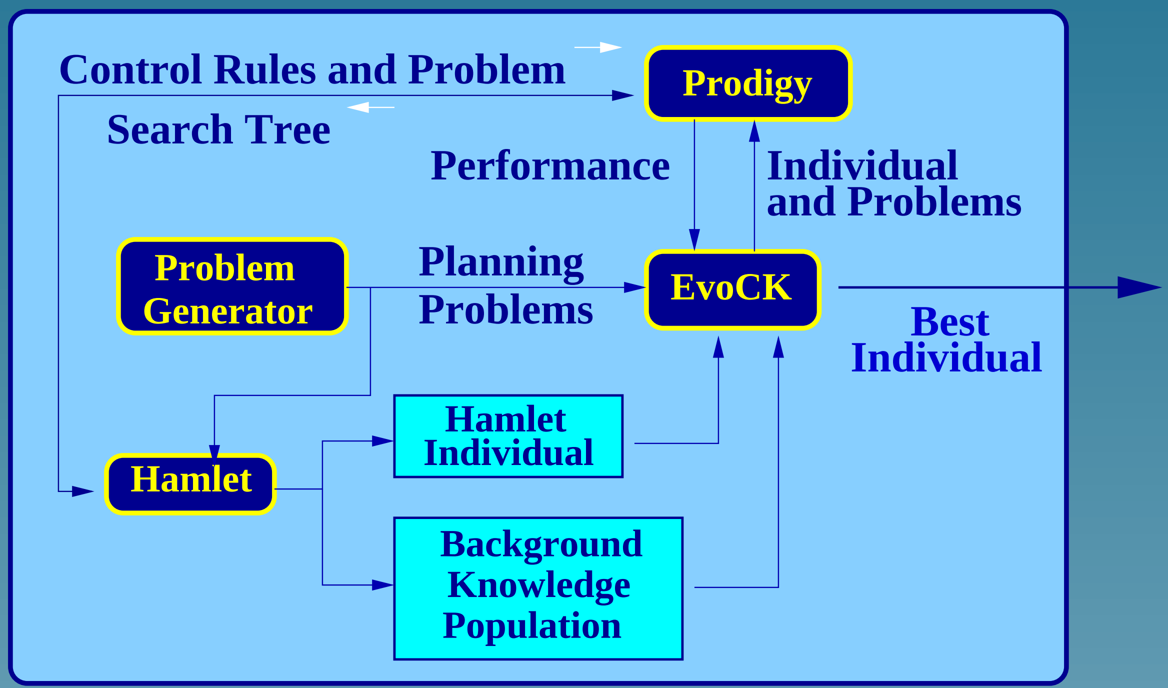
Problems with HAMLET

- It does not always generate better control knowledge by observing more and more examples
  - **incrementality** (partially solved through revisiting problems)
  - generalization and specialization procedures require to **add/delete the right** preconditions
  - it learns from **simple problems** search trees, preferably fully expanded
  - it depends very much on the **training examples** (inductive method): not simple, not difficult (the right examples to learn from might be too difficult)
  - **reduced language** for describing control rules: adding new types of conditions is hard given that generalization/specialization operators are not declaratively represented
- But, it provides a **very good starting point** for another type of learner (machine or human)

Incremental and Non-incremental Learning of Control Knowledge for Planning

1. Motivation
2. Incremental learning. HAMLET
3. **Learning by genetic programming. EVOCK**
4. Discussion

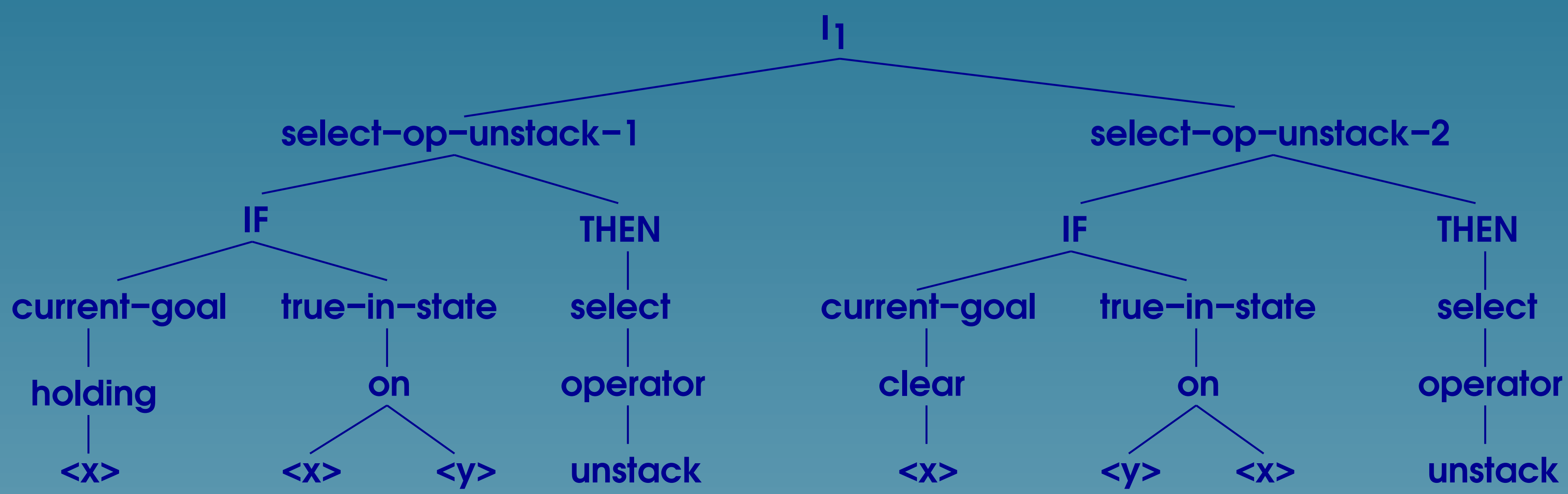
EvoCK architecture



Genetic Programming of control knowledge. EvoCK

- Grammar-based
- Individual: set of control rules
- Genetic operators
  - Crossover (standard, informed, adding)
  - Mutation (standard, removing, adding)
  - Specific (renaming variables, generalization)
- Fitness function
  - Completeness
    - Number of solved problems
    - Number of solved problems better than PRODIGY
  - Generality
  - Compactness

Example of an individual



Learning by genetic programming. EVOCK

Grammar-based GP. Domain-independent

LIST-ROOT-T	$\longrightarrow$	RULE-T | (`list` RULE-T LIST-ROOT-T)
RULE-T	$\longrightarrow$	(`rule` AND-T ACTION-T)
AND-T	$\longrightarrow$	METAPRED-T | (`and` METAPRED-T AND-T)
METAPRED-T	$\longrightarrow$	(`true-in-state` GOAL-T) | (`target-goal` GOAL-T) | (`current-goal` GOAL-T) | (`some-candidate-goals` LIST-OF-GOALS-T)
LIST-OF-GOALS-T	$\longrightarrow$	GOAL-T | (`list-goal` GOAL-T LIST-OF-GOALS-T)
ACTION-T	$\longrightarrow$	(`select-goal` GOAL-T) | (`select-operator` OP-T) | (`select-bindings` BINDINGS-T) | `sub-goal` | `apply`

Learning by genetic programming. EVOCK

Grammar-based GP. Domain-dependent

OP-T	$\longrightarrow$	`load-truck` | `load-airplane` | `unload-truck` | `unload-airplane` | `drive-truck` | `fly-airplane`
BINDINGS-T	$\longrightarrow$	(`load-truck-b` OBJECT-T TRUCK-T LOCATION-T) | (`load-airplane-b` OBJECT-T AIRPLANE-T AIRPORT-T) | (`unload-truck-b` OBJECT-T TRUCK-T LOCATION-T) | (`unload-airplane-b` OBJECT-T AIRPLANE-T AIRPORT-T) | (`drive-truck` TRUCK-T LOCATION-T LOCATION-T) | (`fly-airplane` AIRPLANE-T AIRPORT-T AIRPORT-T)
GOAL-T	$\longrightarrow$	(`at-truck` TRUCK-T LOCATION-T) | (`at-obj` OBJECT-T LOCATION-T) | (`inside-truck` OBJECT-T TRUCK-T) | (`inside-airplane` OBJECT-T AIRPLANE-T)

Incremental and Non-incremental Learning of Control Knowledge for Planning

1. Motivation
2. Incremental learning. HAMLET
3. Learning by genetic programming. EVOCK
4. **Discussion**

Discussion

Related work

- **Linear**: STRIPS [Fikes *et al.*, 1972], Rubik's cube [Korf, 1985], PRODIGY/EBL [Minton, 1988], STATIC [Etzioni, 1993], DYNAMIC [Pérez and Etzioni, 1992], ALPINE [Knoblock, 1991], GRASSHOPER [Leckie and Zukerman, 1998], LEX [Mitchell *et al.*, 1983], ACM [Langley, 1983], LEBL [Tadepalli, 1989], DOLPHIN [Zelle and Mooney, 1993], EXPERIMENTER [Carbonell and Gil, 1990], . . .
- **Nonlinear "classical"**: SOAR [Laird *et al.*, 1986], FAILSAFE [Bhatnagar, 1992], OBSERVE [Wang, 1994], COMPOSER [Gratch and DeJong, 1992], PRIAR [Kambhampati, 1989], SNLP+EBG [Kambhampati and Kedar, 1991], SNLP+EBL [Katukam and Kambhampati, 1994], UCPOP+EBL [Qu and Kambhampati, 1995], QUALITY [Pérez and Carbonell, 1994], SteppingStone [Ruby and Kibler, 1992], UCPOP+FOIL [Estlin and Mooney, 1995], PIPP [Upal and Elio, 1998], PRODIGY/ANALOGY [Veloso, 1994], DerSNLP [Ihrig and Kambhampati, 1996], HAMLET [Borrajo and Veloso, 1997], EVOCK [Aler *et al.*, 2002], ExEL [Reddy and Tadepalli, 1999], . . .

Discussion

More related work

- **Nonlinear "non classical"**: rewrite rules [Ambite *et al.*, 2000, Upal and Elio, 2000], CAMEL [Ilghami *et al.*, 2002], HTN MODELS [Garland *et al.*, 2001], GRAPHPLAN+EBL [Kambhampati, 2000], SATPLAN+FOIL [Huang *et al.*, 2000], generalized policies [Khardon, 1999, Martín and Geffner, 2000], HAP [Vrakas *et al.*, 2003]
- **MDP models:** reinforcement learning [Kaelbling *et al.*, 1996], Q-LEARNING [Watkins and Dayan, 1992], temporal differences [Sutton, 1988, Tesauro, 1992], LOPE [García-Martínez and Borrajo, 2000]

Discussion

HAMLET vs. EVOCK

- HAMLET
  - *knows* about learning in planning
  - learning operators require right examples to modify candidate hypotheses
  - incremental
  - planner and language dependent
- EVOCK
  - does not know it is doing learning in planning
  - learning operators do not require right examples to modify candidate hypotheses
  - non-incremental
  - grammar dependent

Incrementality

- Incrementality allows
  - focusing on one example: juice extraction (EBL)
  - generating the next-best example
  - better approaching changes in target concept (life-long learning)
  - knowing what control rule is responsible for what

Incrementality

- Incrementality allows
  - focusing on one example: juice extraction (EBL)
  - generating the next-best example
  - better approaching changes in target concept (life-long learning)
  - knowing what control rule is responsible for what
- Non-incrementality allows
  - having a global view of a distribution of examples
  - reducing the effect of noise or particular examples
  - better deciding what and how to generalize and specialize

Discussion

Incrementality

- Incrementality allows
  - focusing on one example: juice extraction (EBL)
  - generating the next-best example
  - better approaching changes in target concept (life-long learning)
  - knowing what control rule is responsible for what
- Non-incrementality allows
  - having a global view of a distribution of examples
  - reducing the effect of noise or particular examples
  - better deciding what and how to generalize and specialize

They can be complementary

General discussion

- Learning techniques should reflect somehow the way by which decisions are made by the problem solver **forward vs. backward**
- The knowledge about how to make a decision should be explicit in the meta-state **evaluation functions or cost functions**
- The base problem solver should be able to solve training problems **are easy or incompletely solved problems enough?**
- If quality is important, it should also provide at least two different-quality solutions **all solutions is the optimum**
- If a learning technique acquires individual control knowledge, the decisions should be reproducible to be of use **utility problem**

General discussion

- Learning in problem solving also needs to worry about representativeness of examples **much bigger search spaces**
- It is difficult to add conditions on numbers, negative constrains (and quantification) to the rules **representation**
- Combining machine learning and humans is a very effective approach **mixed initiative, domain axioms, extra predicates, temporal formulae, . . .**

On evaluation of learning in planning

- Difficult task
- What to measure?
  - Efficiency: time, solved problems
  - Quality: solution length (sequential, parallel), makespan, others
  - Combination
- How to compare?
  - with or without prior knowledge
  - domain representation
  - set of problems
- Against what?
  - different learners in different planners
  - knowledge-based planners
  - efficient state of the art planners

Discussion

Still to be solved. Zenotravel

```
(control-rule select-airplane-for-zoom
   (if (current-goal (at-object <object> <airport>))
       (current-operator zoom)
       (true-in-state (at-object <object> <airport1>))
       (true-in-state (at-airplane <plane> <airport2>))
       (cheapest-airplane-for-zoom <object> <plane>
                                   <airport> <airport1>
                                   <airport2>))
   (then select bindings ((<obj> . <object>)
                          (<airplane> . <plane>))))
```

Discussion

More recent and future work

- Mixed initiative
- Effects of knowledge representation
- Effects of prior knowledge
- Learning for multiple criteria
- Learning for HTN+POP planners
- Using numerical predicates on conditions of control rules
- Active learning: on-line generation of appropriate training problems
- Learning for planning and scheduling
- Learning in more recent problem solvers

Referencias

[Aler *et al.*, 2002] Ricardo Aler, Daniel Borrajo, and Pedro Isasi. Using genetic programming to learn and improve control knowledge. *Artificial Intelligence*, 141(1-2):29–56, October 2002.

[Ambite *et al.*, 2000] José Luis Ambite, Craig A. Knoblock, and Steven Minton. Learning plan rewriting rules. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, pages 14–17, Breckenbridge, CO (USA), April 2000.

[Bhatnagar, 1992] Neeraj Bhatnagar. *On-line learning from search failures*. PhD thesis, Rutgers University, 1992.

[Borrajo and Veloso, 1997] Daniel Borrajo and Manuela Veloso. Lazy incremental learning of control knowledge for efficiently obtaining quality plans. *AI Review Journal. Special Issue on Lazy Learning*, 11(1-5):371–405, February 1997. Also in the book "Lazy Learning", David Aha (ed.), Kluwer Academic Publishers, May 1997, ISBN 0-7923-4584-3.

[Carbonell and Gil, 1990] Jaime G. Carbonell and Yolanda Gil. Learning by experimentation: The operator refinement method. In R. S. Michalski and Y. Kodratoff, editors, *Machine Learning: An Artificial Intelligence Approach, Volume III*, pages 191–213. Morgan Kaufmann, Palo Alto, CA, 1990.

[Estlin and Mooney, 1995] Tara A. Estlin and Raymond Mooney. Hybrid learning of search control for partial order planning. Technical report, University of Texas, 1995.

[Etzioni, 1993] Oren Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62(2):255–301, 1993.

[Fikes *et al.*, 1972] Richard E. Fikes, P. E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[García-Martínez and Borrajo, 2000] Ramón García-Martínez and Daniel Borrajo. An integrated approach of learning, planning, and execution. *Journal of Intelligent and Robotic Systems*, 29(1):47–78, September 2000.

[Garland *et al.*, 2001] A. Garland, K. Ryall, and C. Rich. Learning hierarchical task models by defining and refining examples. In *In First International Conference on Knowledge Capture*, 2001.

[Gratch and DeJong, 1992] Jonathan Gratch and Gerald DeJong. COMPOSER: A probabilistic solution to the utility problem in speed-up learning. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 235–240, 1992.

[Huang *et al.*, 2000] Yi-Cheng Huang, Bart Selman, and Henry Kautz. Learning declarative control rules for constraint-based planning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning, ICML'00*, Stanford, CA (USA), June-July 2000.

[Ihrig and Kambhampati, 1996] Laurie H. Ihrig and Subbarao Kambhampati. Design and implementation of a replay framework based on a partial order planner. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 849–854, Portland, Oregon, USA, 1996. AAAI Press / The MIT Press.

[Ilghami *et al.*, 2002] Okhtay Ilghami, Dana S. Nau, Héctor Muñoz-Avila, and David W. Aha. Camel: Learning method preconditions for HTN planning. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS-02)*, pages 131–141, Toulouse (France), 23-17 April 2002. AAAI Press.

[Kaelbling *et al.*, 1996] Lelie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *International Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[Kambhampati and Kedar, 1991] Subbarao Kambhampati and Smadar Kedar. Explanation based generalization of partially ordered plans. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 679–685, Anaheim, CA, 1991. AAAI Press.

[Kambhampati, 1989] Subbarao Kambhampati. *Flexible Reuse and Modification in Hierarchical Planning: A Validation Structure Based Approach*. PhD thesis, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, MD, 1989.

[Kambhampati, 2000] Subbarao Kambhampati. Planning graph as a (dynamic) CSP: Exploiting EBL, DDB and other CSP search techniques in Graphplan. *Journal of Artificial Intelligence Research*, 12:1–34, 2000.

[Katukam and Kambhampati, 1994] Suresh Katukam and Subbarao Kambhampati. Learning explanation-based search control rules for partial order planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 582–587, Seattle, WA, 1994. AAAI Press.

[Khardon, 1999] Roni Khardon. Learning action strategies for planning domains. *Artificial Intelligence*, 113(1-2):125–148, 1999.

[Knoblock, 1991] Craig A. Knoblock. *Automatically Generating Abstractions for Problem Solving*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991. Available as technical report CMU-CS-91-120.

[Korf, 1985] Richard E. Korf. Macro-operators: A weak method for learning. *Artificial Intelligence*, 26:35–77, 1985.

[Laird *et al.*, 1986] John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

[Langley, 1983] Pat Langley. Learning effective search heuristics. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 419–421, Los Altos, CA, 1983. Morgan Kaufmann.

[Leckie and Zukerman, 1998] Christopher Leckie and Ingrid Zukerman. Inductive learning of search control rules for planning. *Artificial Intelligence*, 101(1–2):63–98, May 1998.

[Martín and Geffner, 2000] Mario Martín and Héctor Geffner. Learning generalized policies in planning using concept languages. In *Proceedings of the 7th Int. Conf. on Knowledge Representation and Reasoning (KR 2000)*, Colorado, 2000. Morgan Kaufmann.

[Minton, 1988] Steven Minton. *Learning Effective Search Control Knowledge: An Explanation-Based Approach*. Kluwer Academic Publishers, Boston, MA, 1988.

[Mitchell *et al.*, 1983] Tom M. Mitchell, Paul E. Utgoff, and R. B. Banerji. Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine Learning, An Artificial Intelligence Approach*. Tioga Press, Palo Alto, CA, 1983.

[Pérez and Carbonell, 1994] M. Alicia Pérez and Jaime G. Carbonell. Control knowledge to improve plan quality. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 323–328, Chicago, IL, 1994. AAAI Press, CA.

[Pérez and Etzioni, 1992] M. Alicia Pérez and Oren Etzioni. DYNAMIC: A new role for training problems in EBL. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 367–372. Morgan Kaufmann, Aberdeen, Scotland, 1992.

[Qu and Kambhampati, 1995] Yong Qu and Subbarao Kambhampati. Learning search control rules for plan-space planners: Factors affecting the performance. Technical report, Arizona State University, February 1995.

[Reddy and Tadepalli, 1999] Chandra Reddy and Prasad Tadepalli. Learning horn definitions: Theory and an application to planning. *New Generation Computing*, 17:77–98, 1999.

[Ruby and Kibler, 1992] David Ruby and Dennis Kibler. Learning episodes for optimization. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 379–384, Aberdeen, Scotland, 1992. Morgan Kaufmann.

[Sutton, 1988] Richard Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988.

[Tadepalli, 1989] Prasad Tadepalli. Lazy explanation-based learning: A solution to the intractable theory problem. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 694–700, San Mateo, CA, 1989. Morgan Kaufmann.

[Tesauro, 1992] Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8(3/4):257–277, May 1992.

[Upal and Elio, 1998] Muhammad Afzal Upal and Reneé Elio. Learning to improve quality of the solutions produced by partial-order planners. In *In Notes of the AIPS-98 Workshop on Knowledge Engineering and Acquisition for Planning: Bridging Theory and Practice*, pages 94–104, 1998.

[Upal and Elio, 2000] M. Apzal Upal and Renee Elio. Learning search control rules versus rewrite rules to improve plan quality. In *Proceedings of the Thirteenth Canadian Conference on Artificial Intelligence*, pages 240–253, New York, 2000. Springer-Verlag.

[Veloso, 1994] Manuela Veloso. *Planning and Learning by Analogical Reasoning*. Springer Verlag, December 1994.

[Vrakas *et al.*, 2003] Dimitris Vrakas, Grigorios Tsoumakas, Nick Bassiliades, and Ioannis Vlahavas. Learning rules for adaptive planning. In *Proceedings of ICAPS'03*, Trento (Italia), June 2003.

[Wang, 1994] Xuemei Wang. Learning planning operators by observation and practice. In

Proceedings of the Second International Conference on AI Planning Systems, AIPS-94, pages 335–340, Chicago, IL, June 1994. AAAI Press, CA.

[Watkins and Dayan, 1992] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3/4):279–292, May 1992.

[Zelle and Mooney, 1993] J. Zelle and R. Mooney. Combining FOIL and EBG to speed-up logic programs. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1106–1113, Chambery, France, 1993. Morgan Kaufmann.