Learning to Planning (L2P)

Sungwook Yoon

Subbarao Kambhampati

Arizona State University

http://www.public.asu.edu/~syoon10/L2P-tutorial.html

Outline

- Introduction
- Planning
- Learning Search Control Rules (and Policy)
- Learning Domain Model
- Future Work

Introduction - Disclaimer

- Learning and Planning area has been studied for a long time and one can see that many connections be made from any recent work to prior works
- In this tutorial, we may not cover all the related studies. However, we will attempt to cover all of the important studies in recent years
- You will find more information from prior tutorials
  - http://www-rcf.usc.edu/~skoenig/icaps/icaps04/tutorial1.html
  - http://rakaposhi.eas.asu.edu/ml-summer.html
- We will pick successful L2P studies and look into details

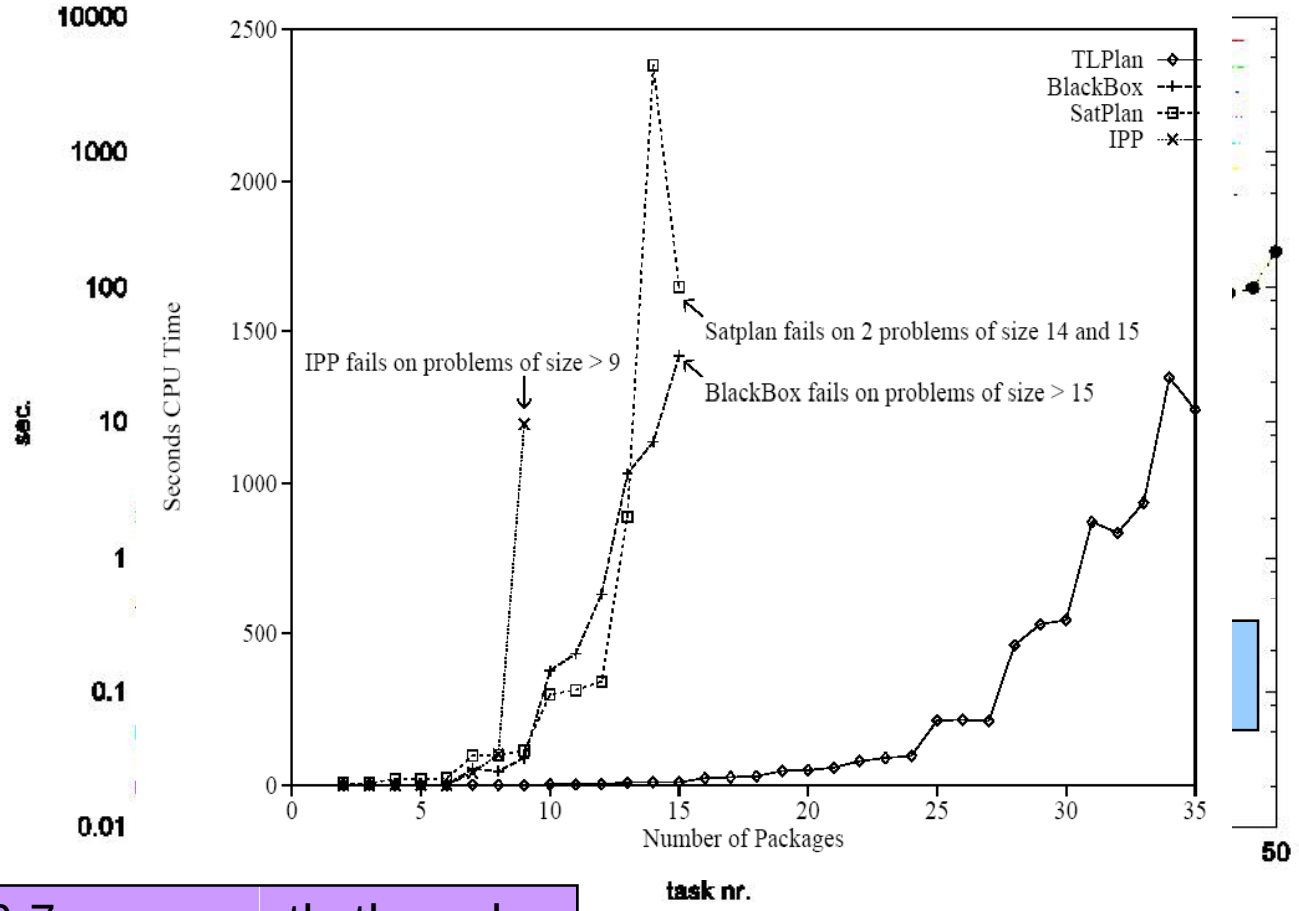
Introduction

- There has been consistent efforts of cross-fertilization between Planning and Learning research areas
- From Planning Side
  - To speed up the solution process
    - Search control
  - To learn more about the target domain
    - Model-lite Planning (Kambhampati, AAAI 2007)
- From Machine Learning Side
  - Challenging Application
    - Planning can be seen as an application of machine learning
    - However, Planning requires sequential decisions, relational structure and the use of potential background knowledge
    - How to get training data?

ARIZONA STATE UNIVERSITY

History of

- Pre-1995 planning algorithms could synthesize about 6 – 10 action plans in minutes
- ➔ Massive dependence on speedup learning techniques
  - ➔ Golden age for Speedup Learning in Planning ☺



- Significant scale-up in the last 6-7 years mostly through powerful reachability heuristics
  - Now, we can synthesize 100 action plans in seconds.
  - Reduced interest in learning as a crutch

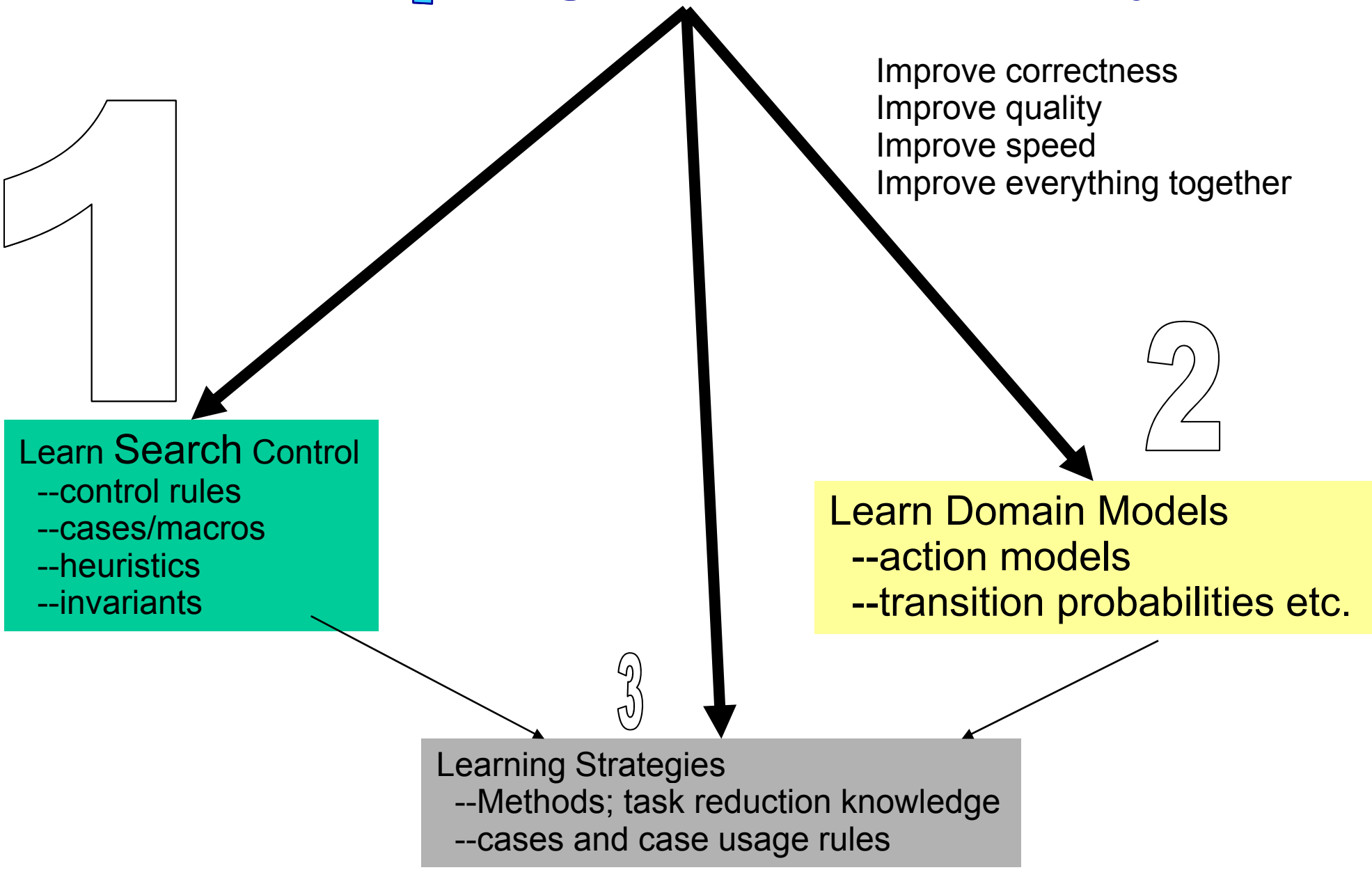
But KBPlanners (customized by humans) did even better

opening up renewed interest in learning the kinds of knowledge humans are are able to put in

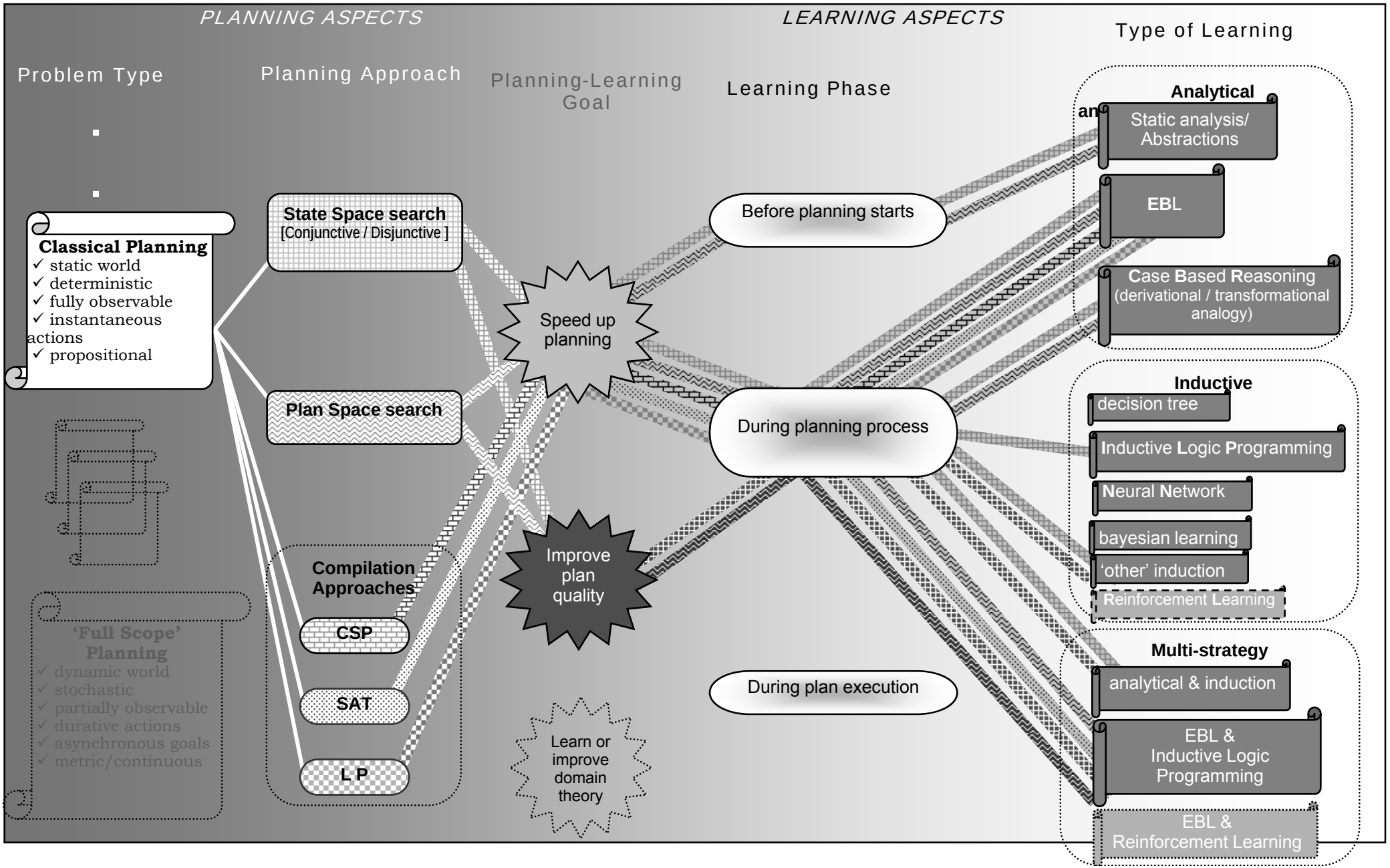
..as well as interest in learning domain models and strategies

Now that we have fast planners, the domain modeling comes to the foreground

Learn to Improve Planning performance



Spectrum of Approaches Tried [AI Mag, 2003]



Introduction – Spectrum of Approaches tried

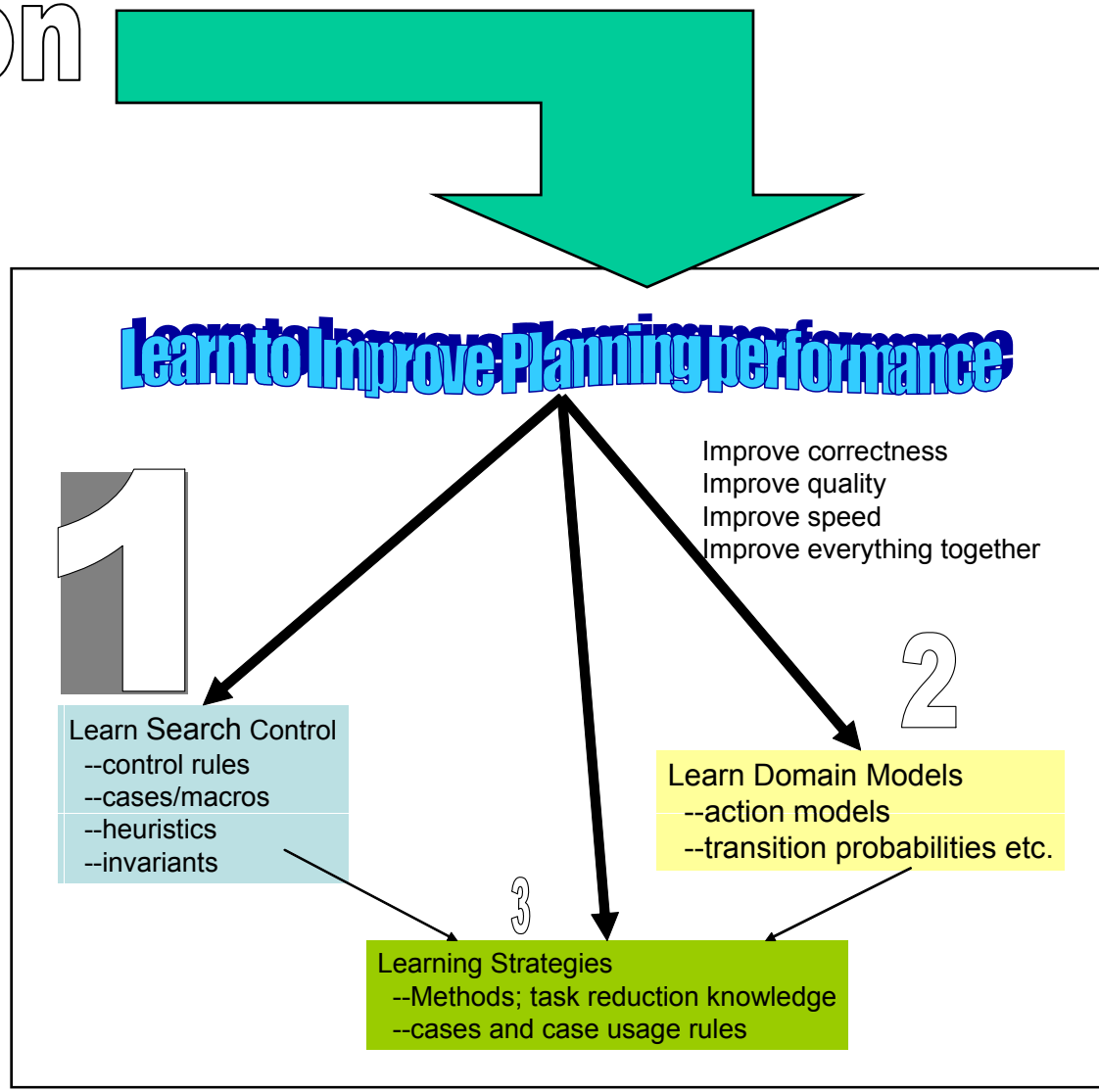
	EBL	CBL	Decision Tree	ILP	Bayesian	Neural Network	Others
State Space Search	Yes (Prodigy)	Yes (Prodigy)	Yes	Yes	Yes (RL)	Yes (FPG)	Yes
Plan Space Search	Yes (SNLP+EBL)	Yes (DerSNLP)		Yes (Dolphin)			
CSP SAT LP				Yes			
HTN				Yes (CAMEL)			Yes (AIPINE)
Domain (operator) Learning				Yes	Yes (RL)		
Modern Planning							Yes (MacroFF)

Introduction – What to watch for

- How the training data is obtained?
  - Self exploration or exercise?
  - From search tree?
  - User provided?
  - Automated planning results?
- How features are generated?
- What kind of background knowledge is needed?
  - We should be positive about providing background knowledge to the learner and planner

Overview

Brief Introduction to Planning



Planning – Overview

- Definition
  - PDDL (STRIPS)
  - KR
- Planning
  - State Space Search
  - Plan Space Search
    - Partial Order Search
    - Heuristic Partial Order Search

Planning – Domain Definition

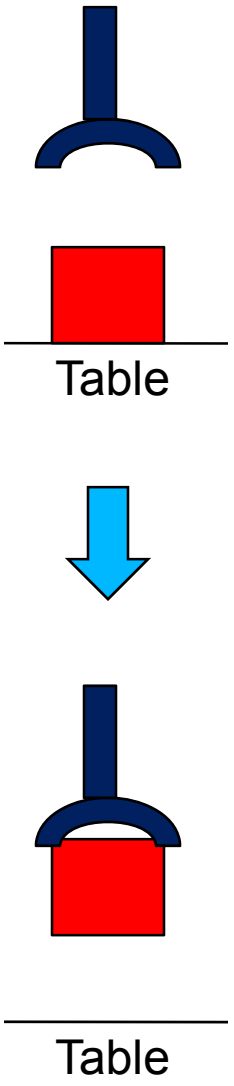
```
(define (domain Blocksworld)
        (:requirements … )
        (:predicates … )

        (:action pickup
                :parameters (?x)
                :precondition (and (clear ?x)
                                   (ontable ?x)
                                   (armempty))
                :effect (and  (holding ?x)
                              (not (clear ?x))
                              (not (ontable ?x))
                              (not (armempty)))
         )
```

Domain Name

:typed :negative-precondition

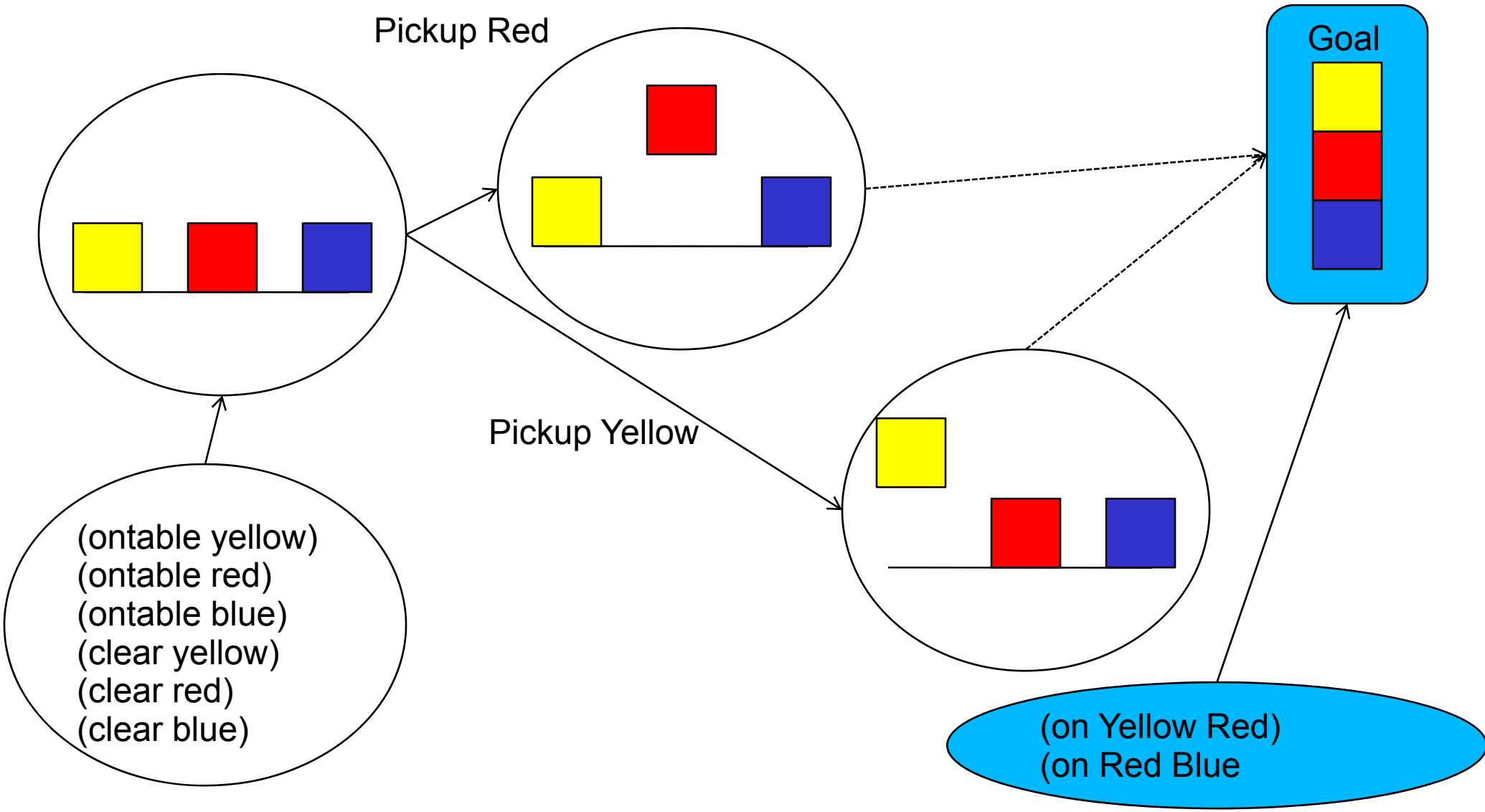
Predicate Definition

Action Definition Schema (name and Parameters) Precondition Effect

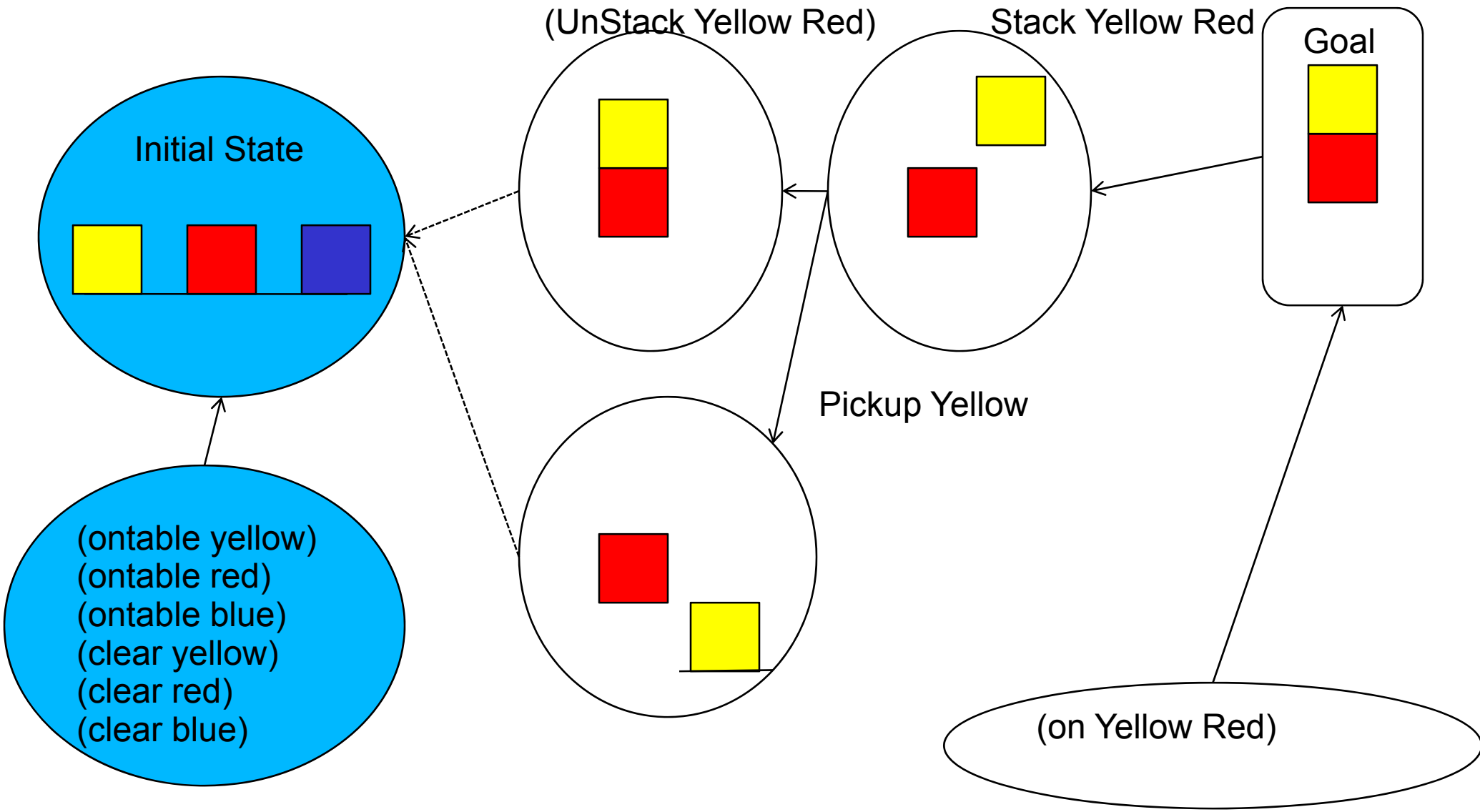


Of course, we need initial state and goal for the problem definition

Planning – Forward State Space Search

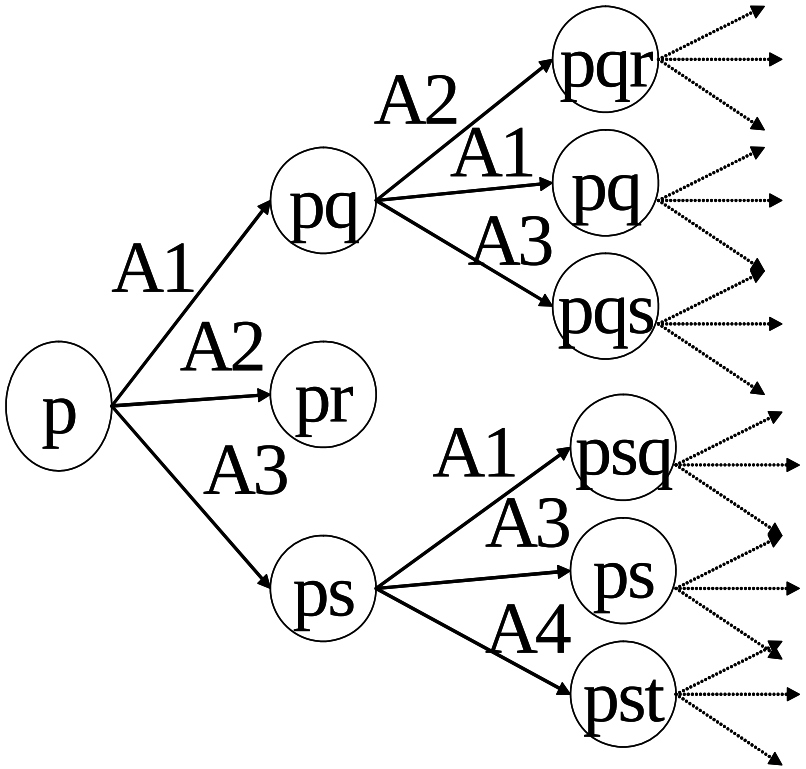


Planning – Backward State Space Search

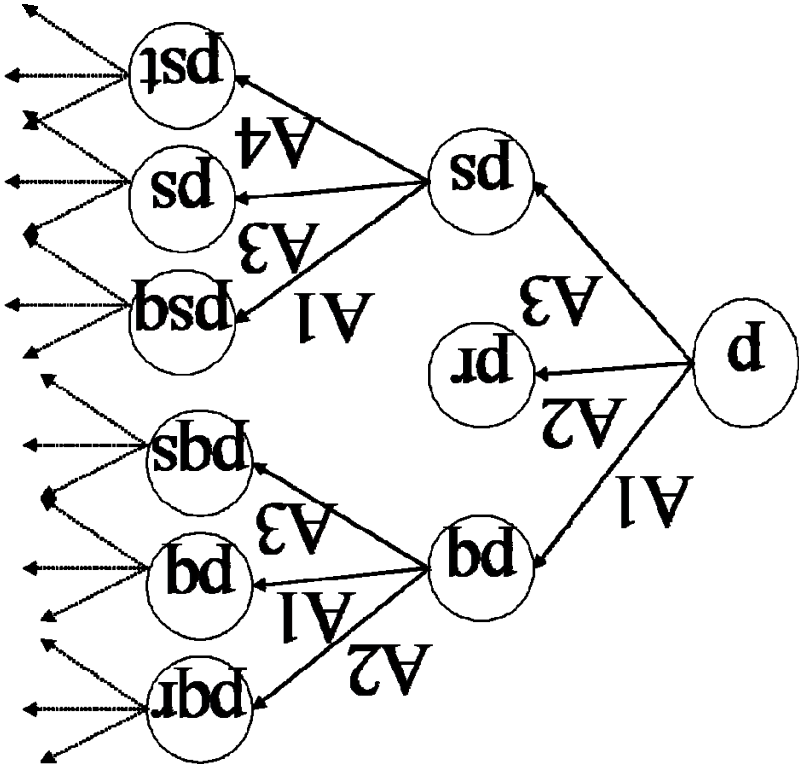


Search & Control

Progression Search



Regression Search

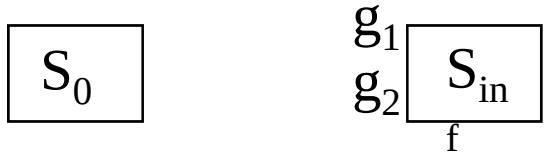


Which branch should we expand?
..depends on which branch is leading (closer) to the goal

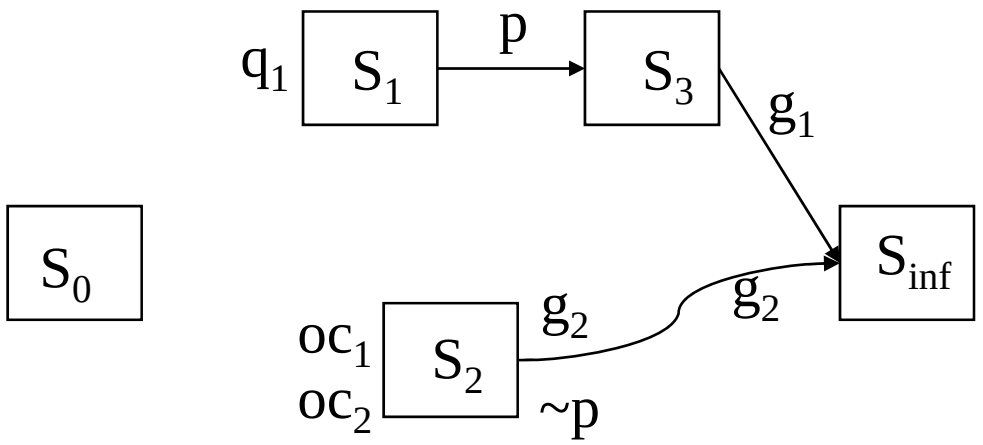
POP Algorithm

1. *Plan Selection*: *Select a plan P from the search queue*
2. *Flaw Selection*: Choose a flaw f (open cond or unsafe link)
3. *Flaw resolution*:
   If f is an open condition,
   choose an action *S* that achieves *f*
   If f is an unsafe link,
   choose promotion or demotion
   Update P
   Return NULL if no resolution exist
4. *If* there is no flaw left, *return* P

1. Initial plan:



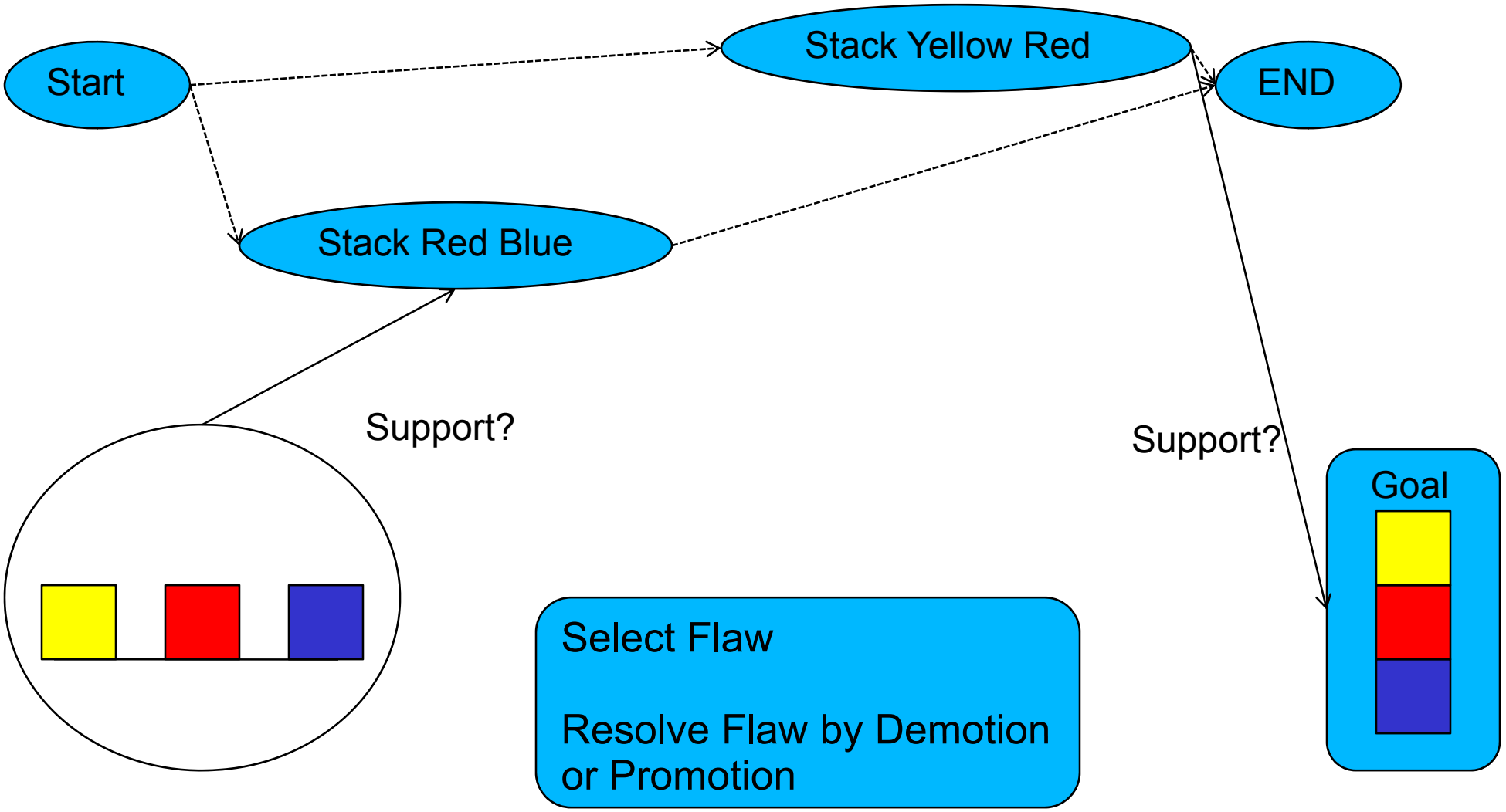
2. Plan refinement (flaw selection and resolution):



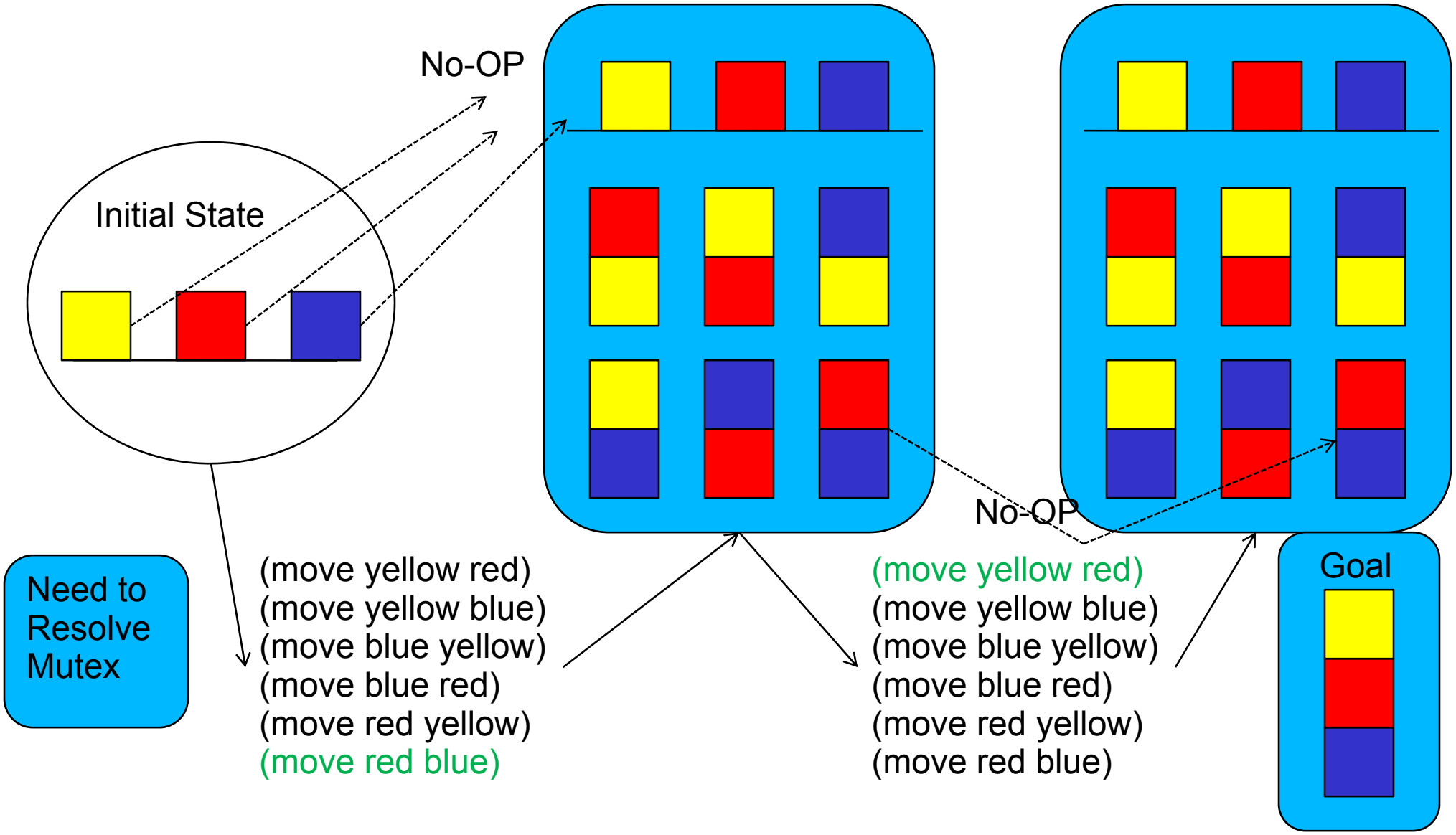
Choice points

- Flaw selection *(open condition? unsafe link? Non-backtrack choice)*
- Flaw resolution/Plan Selection *(how to select (rank) partial plan?)*

Planning – Partial Order Panning



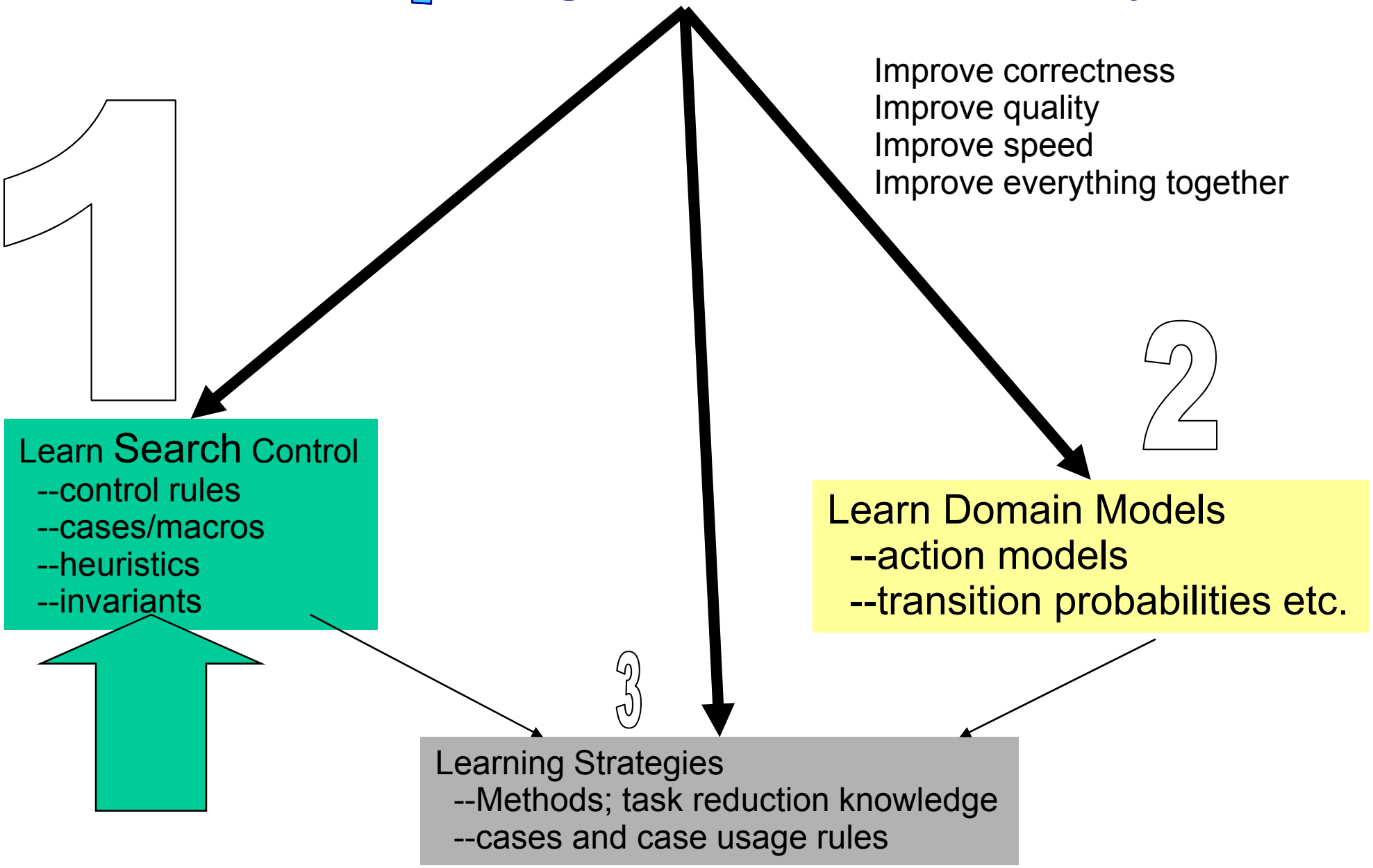
Planning - Graphplan



Planning – Modern Planners

- Forward State Space Search
  - FF
  - YAHSP,MacroFF
- Plan Space Search
  - LPG, SGPlan
- Reduce to other problems
  - Satplan
  - IPPlan

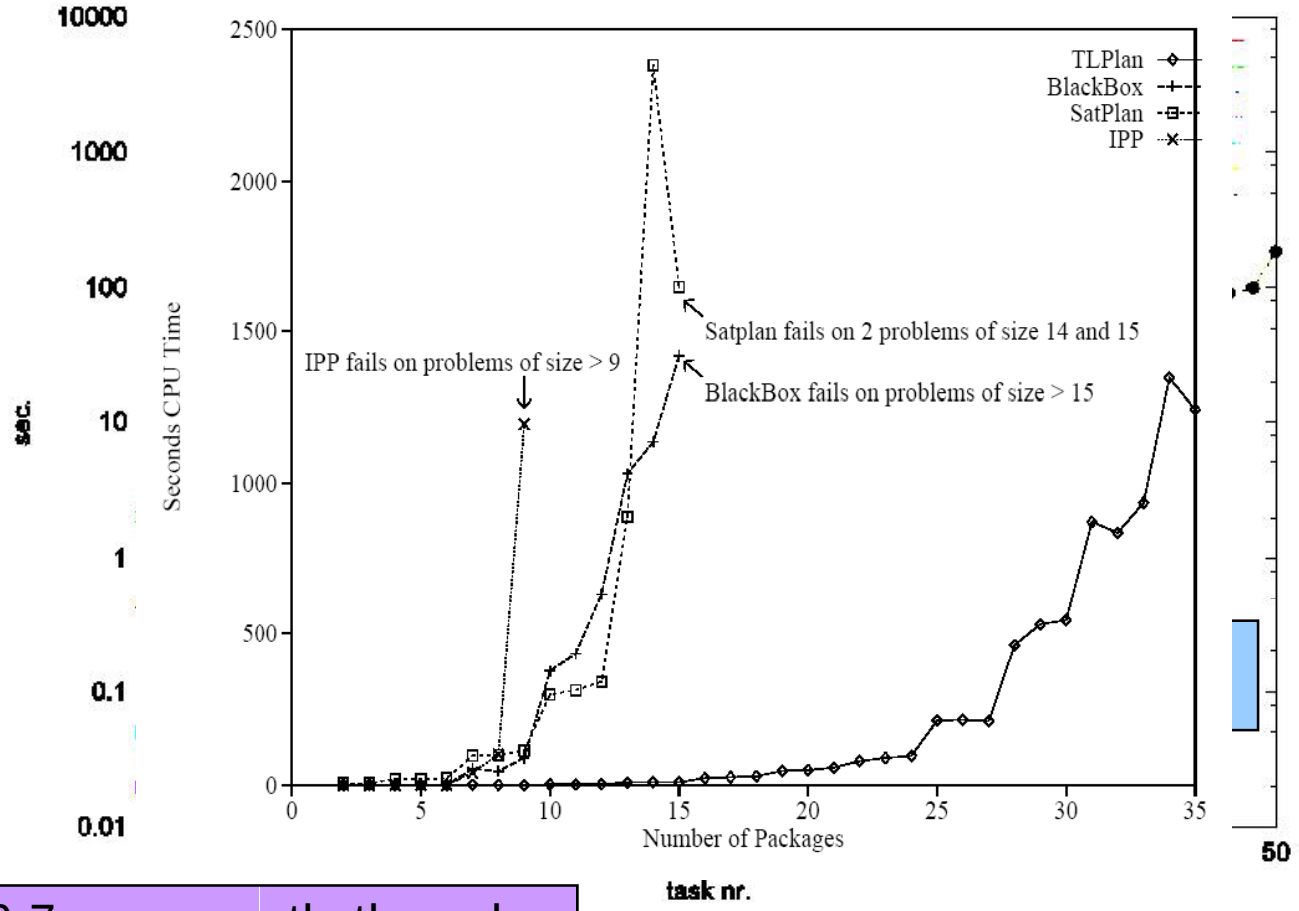
Learn to Improve Planning performance



ARIZONA STATE UNIVERSITY

History of

- Pre-1995 planning algorithms could synthesize about 6 – 10 action plans in minutes
- ➔ Massive dependence on speedup learning techniques
  - ➔ Golden age for Speedup Learning in Planning ☺



- Significant scale-up in the last 6-7 years mostly through powerful reachability heuristics
  - Now, we can synthesize 100 action plans in seconds.
  - Reduced interest in learning as a crutch

But KBPlanners (customized by humans) did even better

opening up renewed interest in learning the kinds of knowledge humans are are able to put in

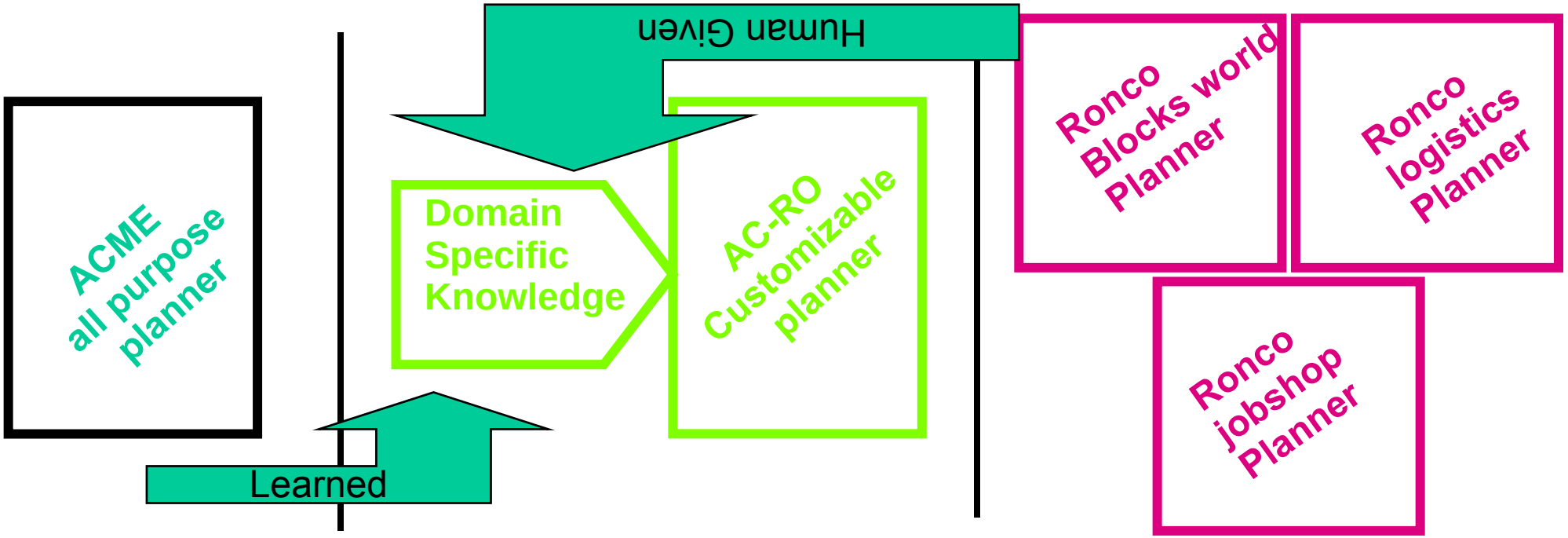
..as well as interest in learning domain models and strategies

Now that we have fast planners, the domain modeling comes to the foreground

Planner Customization (using domain-specific Knowledge)

- Domain independent planners tend to miss the regularities in the domain
- Domain specific planners have to be built from scratch for every domain

An *"Any-Expertise"* Solution: Try adding domain specific control knowledge to the domain-independent planners



How is the Customization Done?

- **Given by humans (often, they are quite willing!)[IPC KBPlanning Track]**
  - **As declarative rules (HTN Schemas, Tlplan rules)**
    - » **Don't need to know how the planner works..**
    - » **Tend to be hard rules rather than soft preferences…**
    - » Whether or not a specific form of knowledge can be exploited by a planner depends on the type of knowledge and the type of planner
  - **As procedures (SHOP)**
    - » **Direct the planner's search alternative by alternative..**

- Through Machine Learning
  - Learning Search Control rules
    - UCPOP+EBL,
    - PRODIGY+EBL, (Graphplan+EBL)
  - Case-based planning (plan reuse)
    - DerSNLP, Prodigy/Analogy
  - Learning/Adjusting heuristics
  - Domain pre-processing
    - Invariant detection; Relevance detection;

      Choice elimination, Type analysis
    - STAN/TIM, DISCOPLAN etc.
    - RIFO; ONLP
  - Abstraction
    - ALPINE; ABSTRIPS, STAN/TIM etc.

We will start with KB-Planning track to get a feel for what control knowledge has been found to be most useful; and see how to get it..

L2P – Search Control

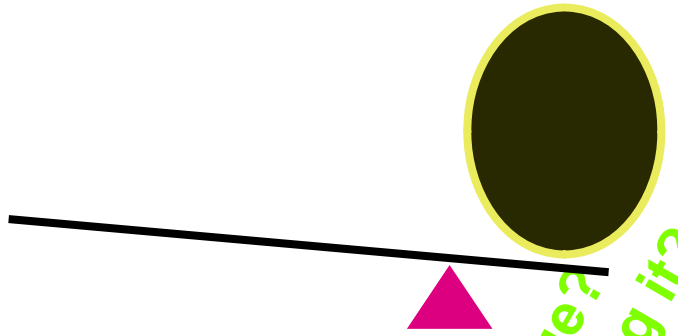
- Human generated search control knowledge
  - Hierarchical structure
    - Shop
      - No effective learner yet (we are getting closer though, HDL and ICARUS, etc.)
    - Macro action
      - MacroFF, Marvin
  - Search control knowledge
    - Constraints on SAT problems
      - Satplan, (Discoplan: automated generation)
        - Huang, Selman and Kautz, ICML 2000
    - Temporal logic
      - Tlplan, TALplan
        - No effective learner yet
    - Precondition of Actions
      - Action selection or rejection rules
      - Many of the Explanation Based Learning Related works (by Katukam and Kambhampati)
      - Yoon,Fern and Givan, UAI 2002, Khardon '99, Martin and Geffner KR 2000

L2P – Search Control

- Other forms of Control Knowledge learning that has been studied
  - Learning Heuristic Function
  - Automated Domain Specific Planner
    - API
  - Learn Cases
  - Relational Reinforcement Learning
  - Learning Measures of Progress

With right domain knowledge any level of performance can be achieved...

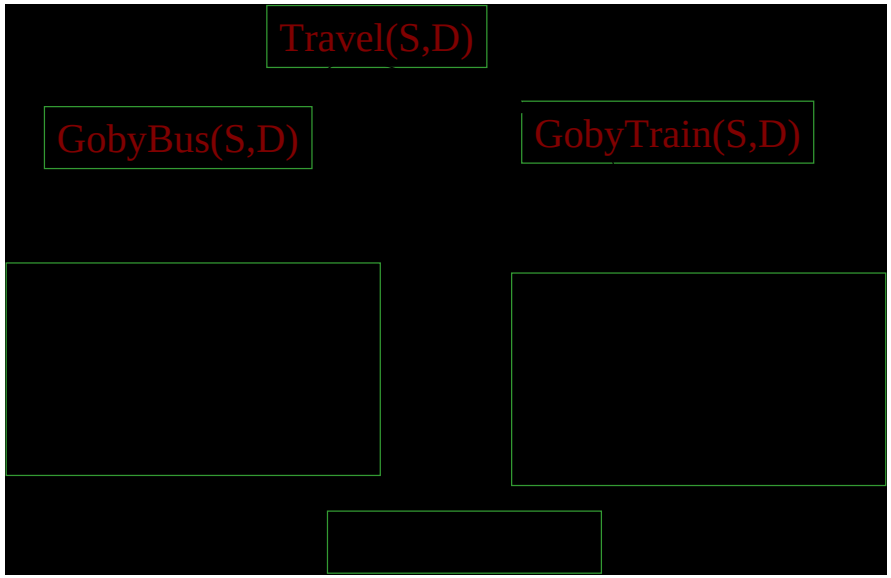
- HTN-SAT, SATPLAN+DOM beat SATPLAN…
  - Expect reduction schemas, declarative knowledge about inoptimal plans
- TLPLAN beats SATPLAN, GRAPHPLAN
  - But expects quite detailed domain knowledge on expected state sequences
- SHOP beats TLPLAN…(but not TALPlan)
  - Expects user to write a “program” for the domain in its language
    - Explicit instructions on the order in which schemas are considered and concatenated



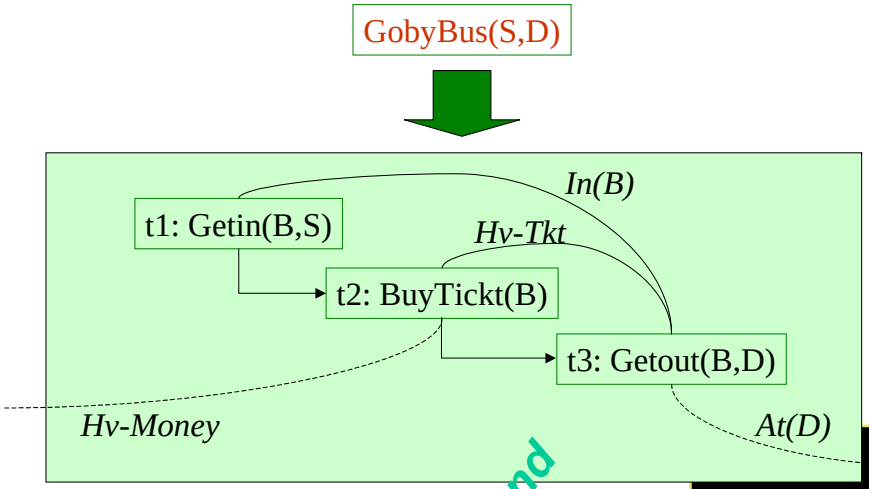
Who gets the credit?
-The provider of domain knowledge?
-The planner that is capable of using it?

Task Decomposition (HTN) Planning

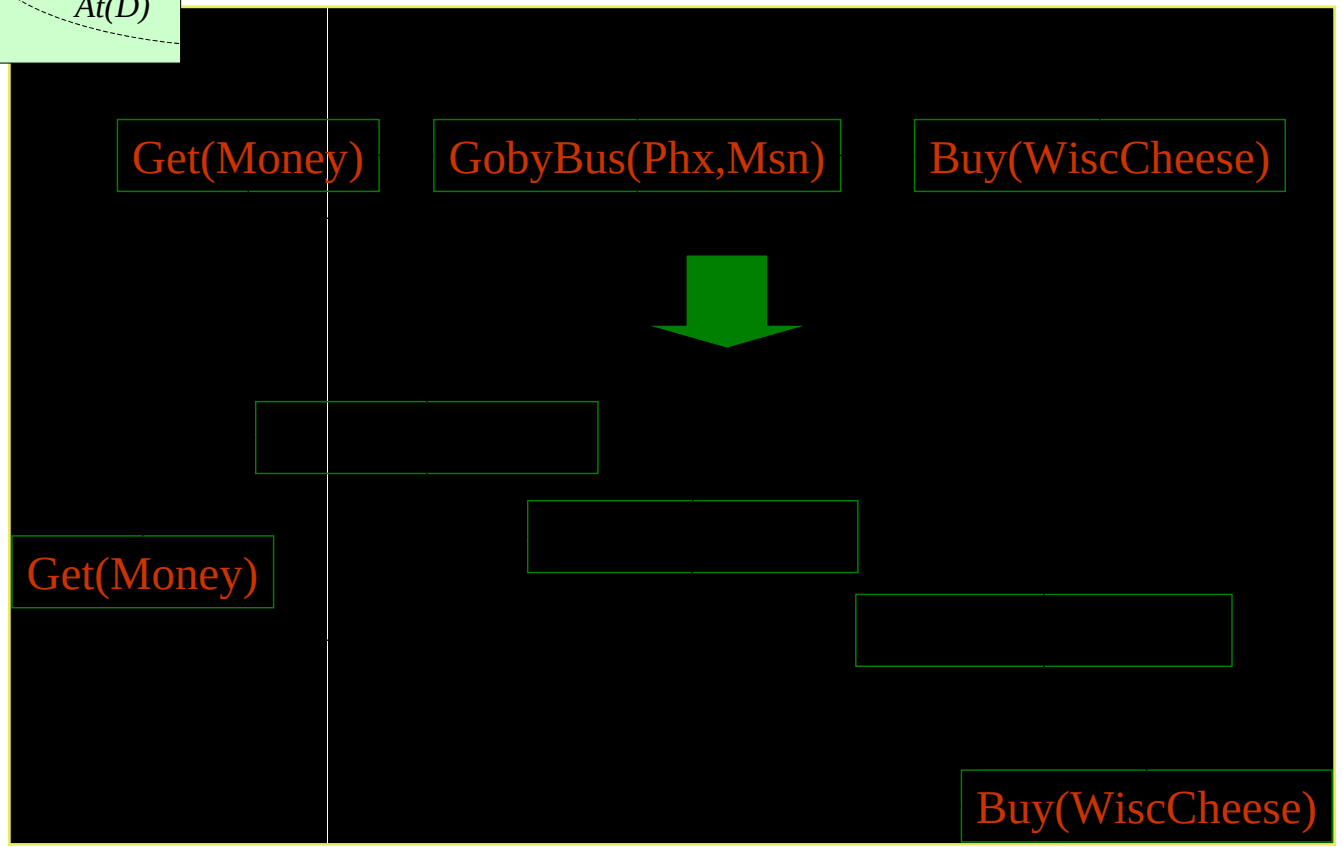
- The OLDEST approach for providing domain-specific knowledge
  - Most of the fielded applications use HTN planning
- Domain model contains non-primitive actions, and schemas for reducing them
- Reduction schemas are given by the designer
  - Can be seen as encoding user-intent
    - *Popularity of HTN approaches a testament of ease with which these schemas are available?*
- Two notions of completeness:
  - Schema completeness
    - (Partial Hierarchicalization)
  - Planner completeness



Modeling Action Reduction



Affinity between reduction schemas and plan-space planning



Dual views of HTN planning

- Capturing hierarchical structure of the domain
  - Motivates top-down planning
    - Start with abstract plans, and reduce them
- Many technical headaches
  - Respecting user-intent, maintaining systematicity and minimality

    [Kambhampati et. al. AAAI-98]
    - Phantomization, filters, promiscuity, downward-unlinearizability..

- Capturing expert advice about desirable solutions
  - Motivates bottom-up planning
    - Ensure that each partial plan being considered is "legal" with respect to the reduction schemas
    - Directly usable with disjunctive planning approaches

      [Mali & Kambhampati, 98]
- Connection to efficiency is not obvious

Relative advantages are still unclear...

[Barrett, 97]

ARIZONA STATE UNIVERSITY

[Nau et al., 99]

Full procedural control: The SHOP way

Shop provides a "high-level" programming language in which the user can code his/her domain specific planner

-- Similarities to HTN planning
-- Not declarative (?)

The SHOP engine can be seen as an interpreter for this language

```
(:method (travel-to ?y)
     (:first (at ?x)
      (at-taxi-stand ?t ?x)
      (distance ?x ?y ?d)
      (have-taxi-fare ?d))
    `((!hail ?t ?x) (!ride ?t ?x ?y)
      (pay-driver ,(+ 1.50 ?d)))
     ((at ?x) (bus-route ?bus ?x ?y))
    `((!wait-for ?bus ?x)
      (pay-driver 1.00)
      (!ride ?bus ?x ?y)))
```

Travel by bus only if going by taxi doesn't work out

Blurs the domain-specific/domain-independent divide

How often does one have this level of knowledge about a domain?

L2P – Search Control - HTN

- If a task can be successfully decomposed into primitive actions, then we can plan
- Learning Systems
  - CaMel learns precondition of Tasks
  - HDL learns methods (including precondition, using version space)
- Structure learning is hard
  - First, how we get label of the abstract actions?
  - Similar to language parsing problem. Means HTN learning need to resolve recursive structure
- Future Approaches
  - Develop compact representation that is suitable for learning systems
  - Labeling should come from some human or automated program

Rules on desirable State Sequences: TLPlan approach

TLPlan [Bacchus & Kabanza, 95/98] controls a forward state-space planner

Rules are written on state sequences using the linear temporal logic (LTL)

LTL is an extension of prop logic with temporal modalities

U	**until**	**[]**	**always**
O	**next**	**<>**	**eventually**

Example:

If you achieve on(B,A), then preserve it until On(C,B) is achieved:

[] (on(B,A) => on(B,A) U on(C,B))

TLPLAN Rules can get quite baroque

Good towers are those that do not violate any goal conditions

$$goodtower(x) \triangleq clear(x) \wedge goodtowerbelow(x)$$

$$goodtowerbelow(x) \triangleq (ontable(x) \wedge \neg \mathrm{GOAL}(\exists[y{:}on(x,y)] \vee holding(x)))$$
$$\vee \exists[y{:}on(x,y)]\, \neg \mathrm{GOAL}(ontable(x) \vee holding(x)) \wedge \neg \mathrm{GOAL}(clear(y))$$
$$\wedge \forall[z{:}\mathrm{GOAL}(on(x,z))]\, z = y \wedge \forall[z{:}\mathrm{GOAL}(on(z,y))]\, z = x$$
$$\wedge goodtowerbelow(y)$$

Keep growing "good" towers, and avoid "bad" towers

$$\Box\Big(\forall[x{:}clear(x)]\, goodtower(x) \Rightarrow \bigcirc goodtowerabove(x)$$
$$\wedge badtower(x) \Rightarrow \bigcirc(\neg\exists[y{:}on(y,x)]\,)$$
$$\wedge (ontable(x) \wedge \exists[y{:}\mathrm{GOAL}(on(x,y))]\, \neg goodtower(y))$$
$$\Rightarrow \bigcirc(\neg holding(x))\Big)$$

How "Obvious" are these rules? Can these be learned?

The heart of TLPlan is the ability to *incrementally* and *effectively* evaluate the truth of LTL formulas.

What are the lessons of KB Track?

- If TLPlan did better than SHOP in ICP, then how are we supposed to interpret it?
  - That TLPlan is a superior planning technology over SHOP?
  - That the naturally available domain knowledge in the competition domains is easier to encode as linear temporal logic statements on state sequences than as procedures in the SHOP language?
  - That Fahiem Bacchus and Jonas Kvarnstrom are way better at coming up with domain knowledge for blocks world (and other competition domains) than Dana Nau?

May be we should "learn" this guidance

Approaches for Learning Search Control

"speedup learning"

Improve an existing planner

Learn "from scratch" how to plan

--Learn "reactive policies"
State x Goal→action

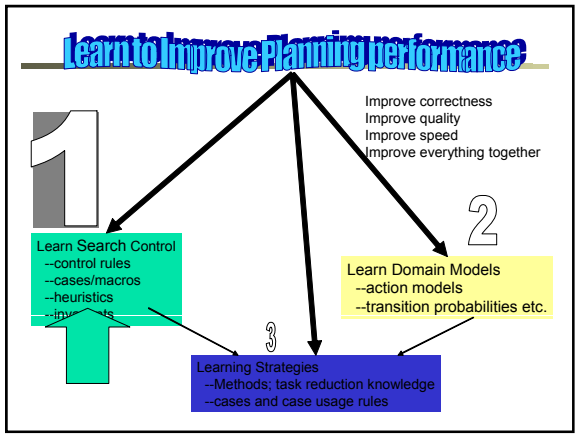
[Work by Khadron, 99;
Givan, Fern, Yoon, 2003→]

Learn rules
to guide choice points

Learn plans
to reuse

--Macros
--Annotated cases

Learn adjustments to
heuristics



No "from-scratch" learner ever placed well in the Intl. Planning Competition.
Macro-FF, an extension of a successful planner called FF, placed 1st in 3 domains in IPC-2004 (..but there was no Knowledge-based Planning track in 2004)

General Strategy for Inductive Learning of Search Control

- Convert to "classification" learning
  - +ve examples: Search nodes on the success path
  - -ve examples: Search nodes one step away from the success path
  - Learn a classifier
    - Classifier may depend on the features of the problem (Init, Goal), as well as the current state.
- Several systems:
  - Grasshopper (Leckie & Zuckerman; 1998)
  - Inductive Logic Programming; (Estlin & Mooney; 1993)

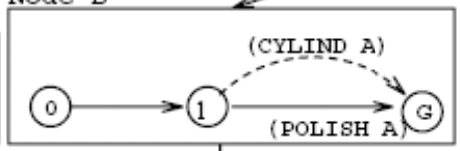
Rule:
Reject stepaddition(ROLL(?X) P G)
If open-cond((POLISH ?X) G)
not-intially-true(POLISH ?X)
(?X = A)

Node A
INITIAL STATE: (0 (COOL A))
GOAL STATE: ((CYLINDRICAL A) G) ((POLISH A) G)

step-addition(ROLL (CYLINDRICAL A) G)

step-addition(LATHE (CYLINDRICAL A) G)
SUCCESS

Node B

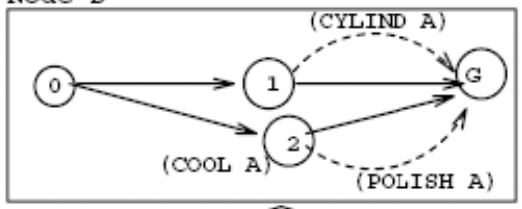


Reason:
(0 < 1) (1 < G)
open-cond((POLISH ?X) G)
has-effect(1 !(COOL ?X))
has-effect(1 !(POLISH ?X))
not-initially-true(POLISH ?X)
(?X = A)

Generalized Rule:
Reject stepaddition(ROLL(?X) P S)
if open-cond((POLISH ?X) S)
not-initially-true(POLISH ?X)

step-addition(POLISH (POLISH A) G)

Node D

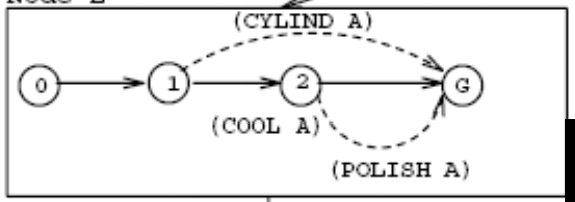


Reason:
(0 < 1)(1 < G)
establishes(2 (POLISH ?X) G)
open-cond((COOL ?X) 2)
has-effect(1 !(COOL ?X))
has-effect(1 !(POLISH ?X))
(?X = A)

demotion((2 (POLISH A) G) 1)

promotion((2 (POLISH A) G) 1)

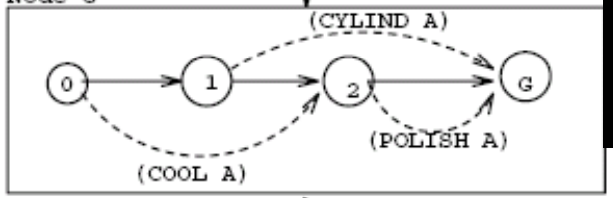
Node E



Node F FAIL
Reason:(1 < G)(G < 1)

establishment(0 (COOL A) G)

Node G



has-effect(1 !(COOL ?X))
(?X = A)

demotion((0 (COOL A) 2) 1)

promotion((0 (COOL A) 2) 1)

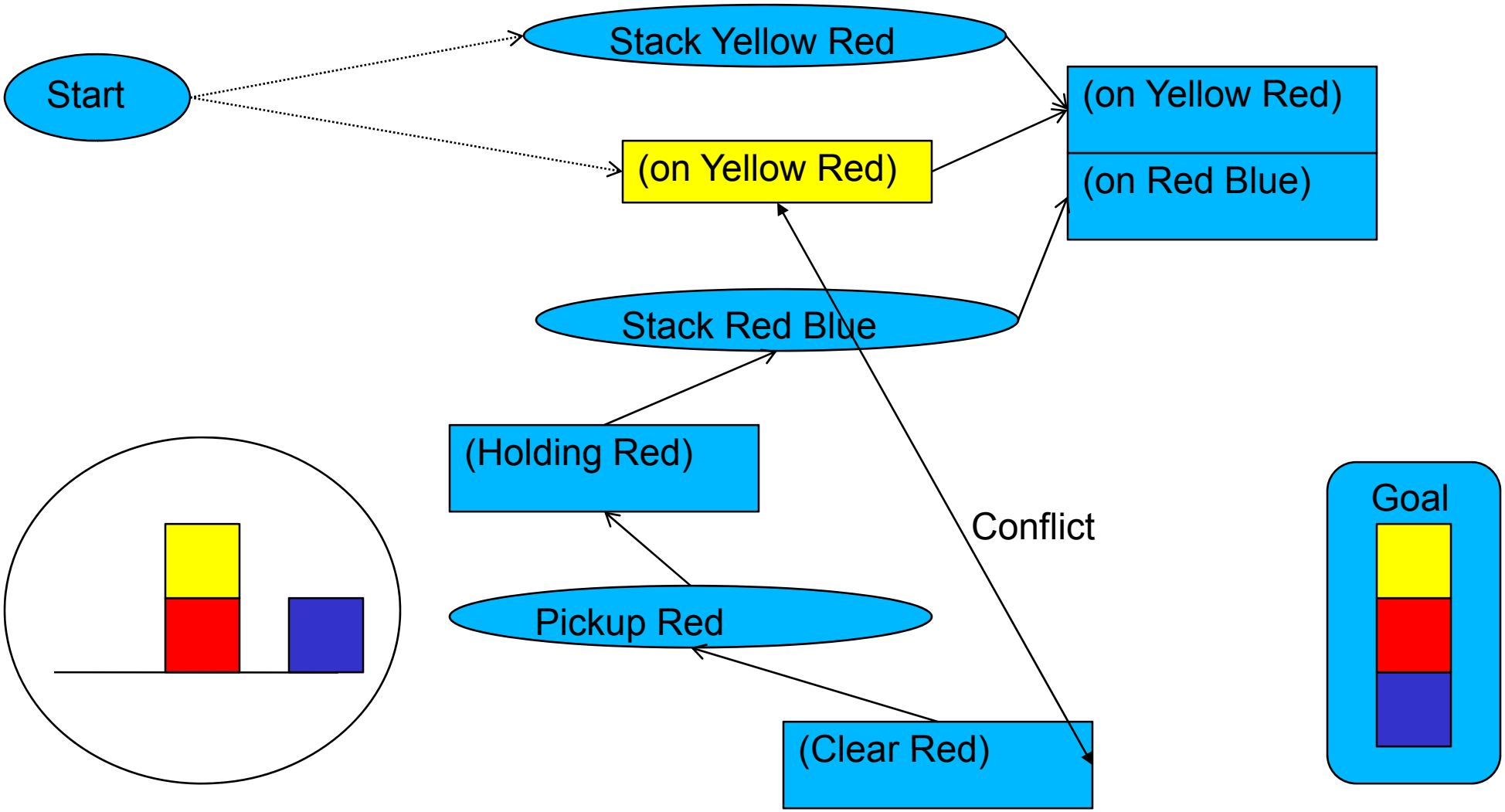
Node H Fail
Reason:(0 < 1)(1 < 0)
(initial explanation)

Node I Fail
Reason:(1 < 2)(2 < 1)
(initial explanation)

If Polished(x)@S &
~Initially-True(Polished(x))
Then
REJECT
Stepadd(Roll(x),Cylindrical(x)@s)

= : codesignates
! : Negation
S : any step
?x : any object
cylind : cylindrical

L2P – Search Control - EBL



L2P – Search Control - EBL

- Explain the failure
  - Explanation process can be applied to any form of training example
  - In the previous example, explain why the initial fact (on Yellow Red) cannot support the goal directly
    - (goal (on Red Blue))
    - (needed (clear Red))
    - (on Yellow Red) -> (not (clear Red)) From Background knowledge
- Generalize the explanation
  - (goal (on ?x ?y)) and (needed (clear ?x)) and (established (on ?z ?x)) -> do not use (on ?z ?x) to establish another goal
- Store the learned rules and use them later

Explanation-based Learning

- Start with a labeled example, and some background domain theory
- *Explain*, using the background theory, why the example deserves the label
  - Think of explanation as a way of picking class-relevant features with the help of the background knowledge
- Use the explanation to generalize the example (so you have a general rule to predict the label)
- Used extensively in planning
  - *Given a correct plan for an initial and goal state pair, learn a general plan*
  - *Given a search tree with failing subtrees, learn rules that can predict failures*
  - *Given a stored plan and the situations where it could not be extended, learn rules to predict applicability of the plan*

Learning Search Control Rules

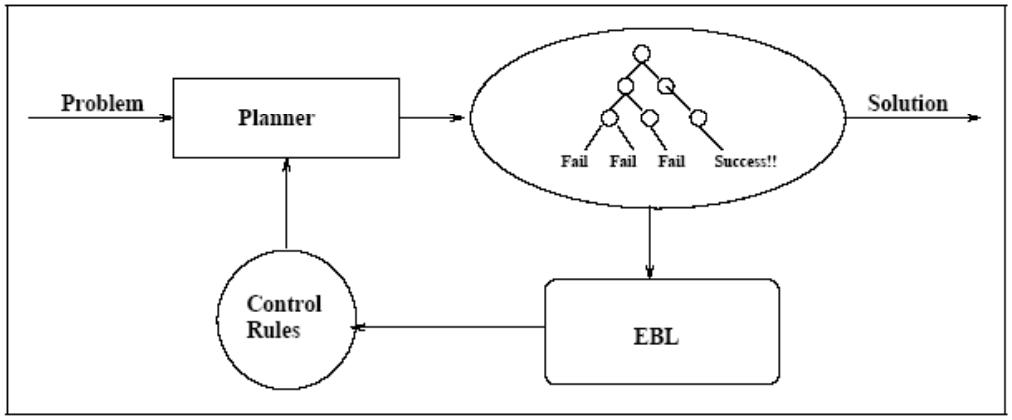
Explain leaf level failures

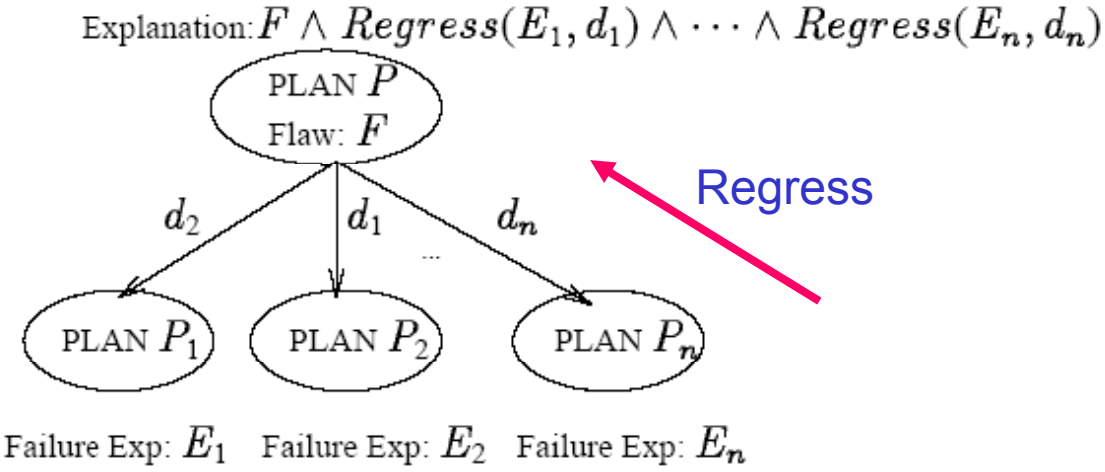
Regress the explanations to compute interior node failure explanations

Use failure explanations to set up control rules

Problems:

- **Most branches end in depth-limits**
  - >No analytical explanation
  - >Use preference rules?
- **The utility problem**
  - >Learn general rules
  - >Keep usage statistics & prune useless rules

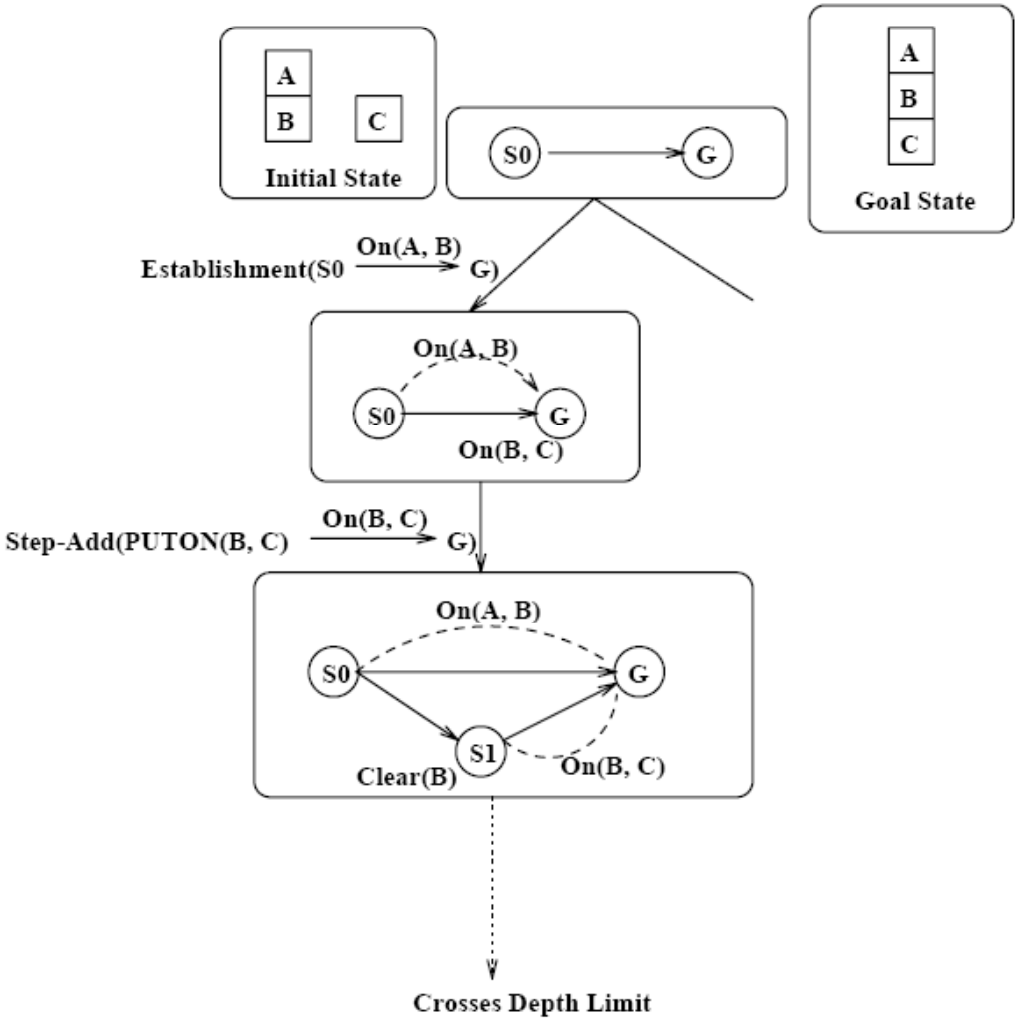




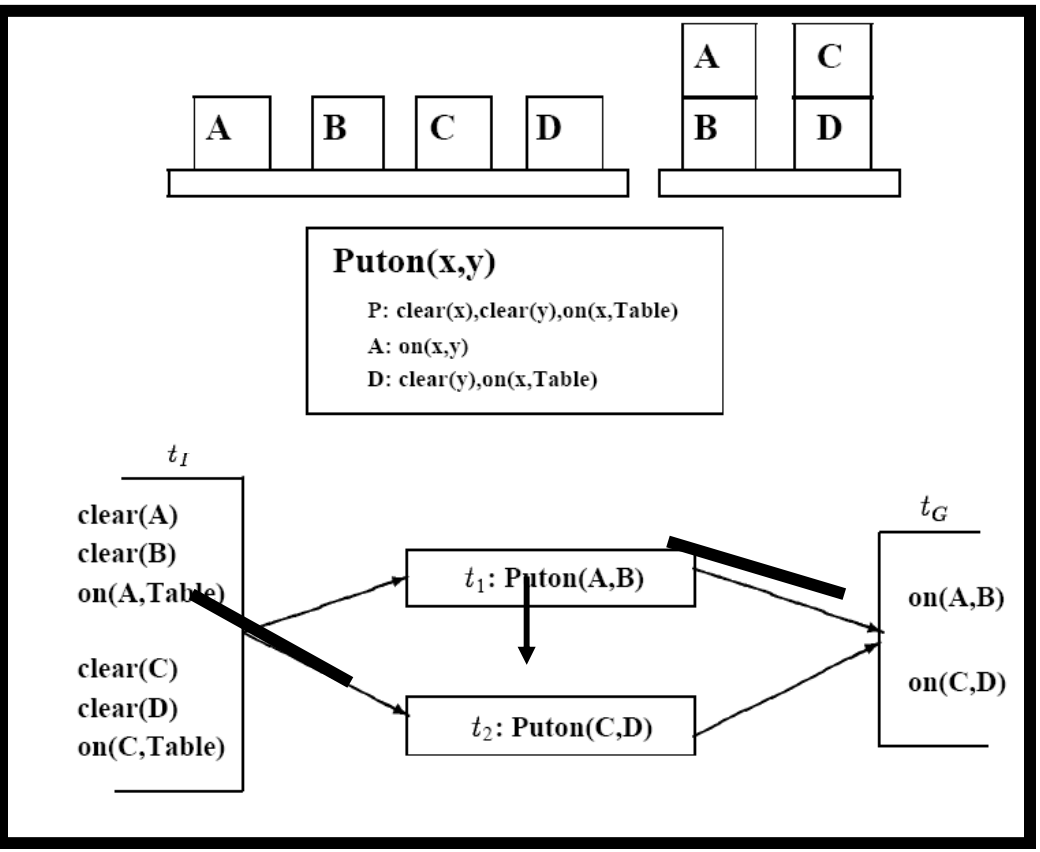
(Kambhampati, Katukam, Qu, 95)

Issues in EBL for Search Control Rules

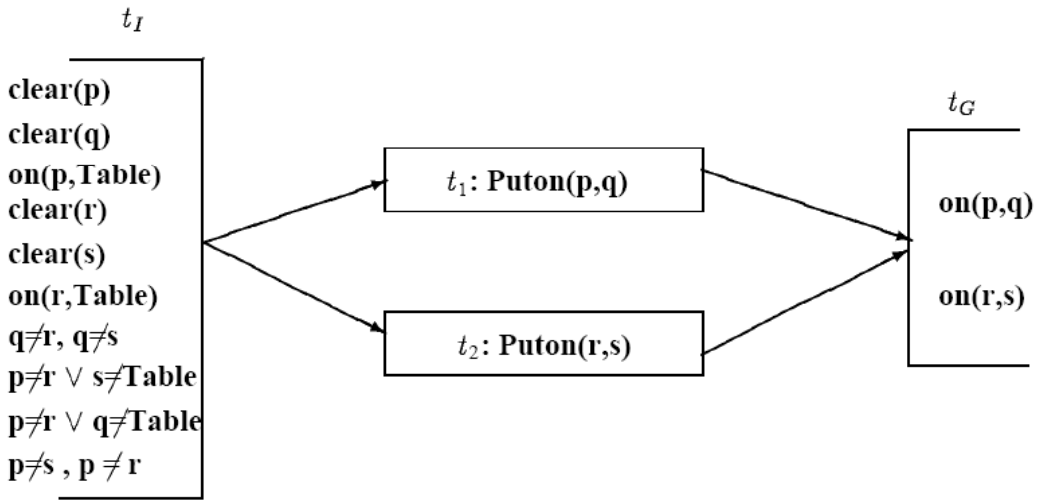
- Effectiveness of learning depends on the explanation
  - Primitive explanations of failure may involve constraints that are directly inconsistent
  - But it would be better if we can unearth hidden inconsistencies
- ..an open issue is to learn with probably incorrect explanations
  - UCPOP+CFEBL



We can also explain (& generalize) Success



Success explanations tend
to involve more components
of the plan than failure explanations



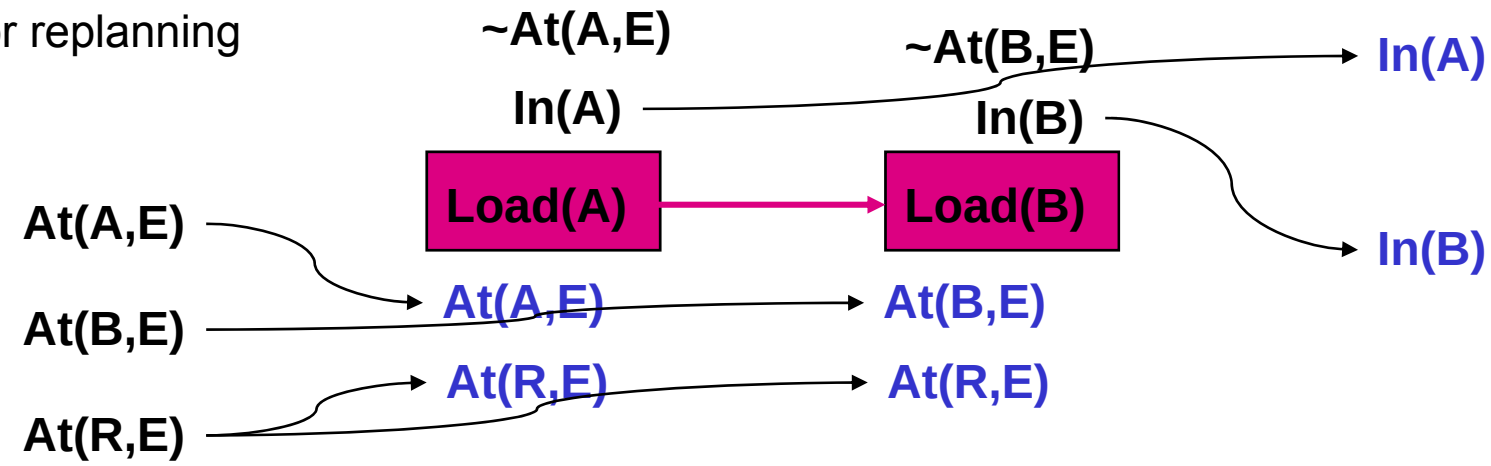
Checking correctness of a plan: The Causal Approach

- Causal Proof: Check if each of the goals and preconditions of the action are
  - "established" : There is a preceding step that gives it
  - "unclobbered": No possibly intervening step deletes it
    - Or for every preceding step that deletes it, there exists another step that precedes the conditions and follows the deleter adds it back.

white Knight

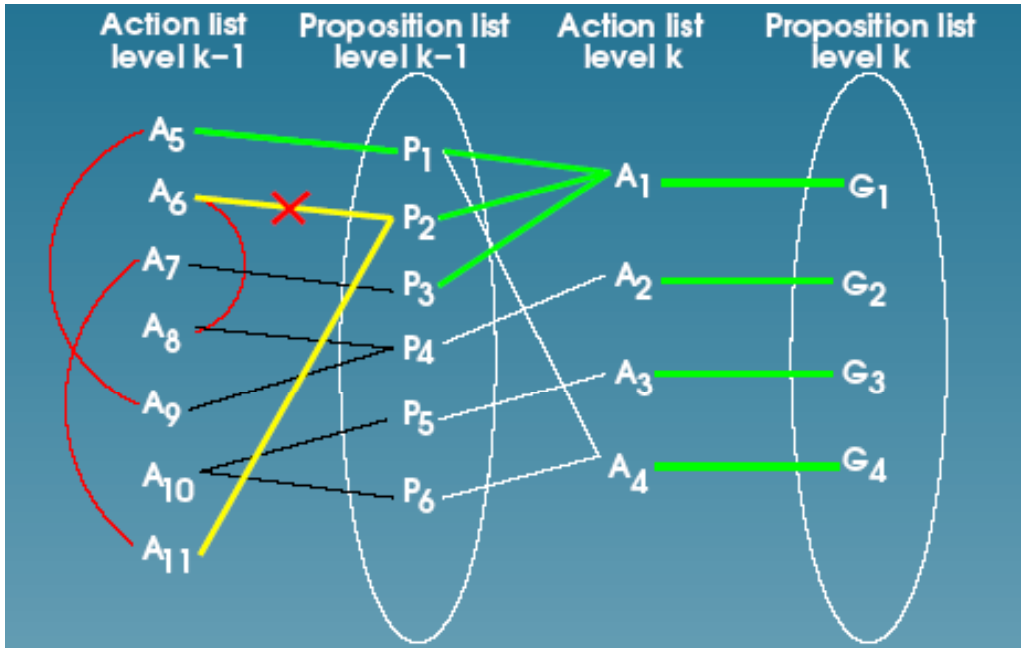
Causal proof is "More like the way we reason with plans"

- "local" (checks correctness one condition at a time)
- "state-less" (does not need to know the states preceding actions)
  - Easy to extend to *durative* actions
- "incremental" with respect to action insertion
  - Great for replanning



Status of EBL learning in Planning

- Explanation-based learning from failures has been ported to modern planners
  - GP-EBL [Kambhampati, 2000] ports EBL to Graphplan
    - "Mutual exclusion relations" are learned
      - (exploits the connection between EBL and "nogood" learning in CSP)
    - Impressive speed improvements
      - EBL is considered standard part of Graphplan implementation now..
    - …but much of the learning was *intra problem*



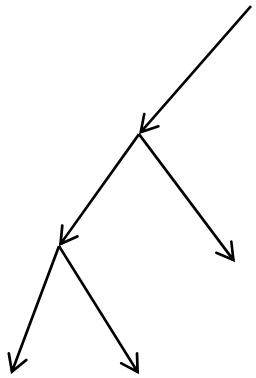
Some misconceptions about EBL

- Misconception 1: EBL *needs* complete and correct background knowledge
    - (Confounds “Inductive vs. Analytical” with “Knowledge rich vs. Knowledge poor”)
  - *If you have complete and correct knowledge then the learned knowledge will be in the deductive closure* of the original knowledge;
  - If not, then the learned knowledge will be tentative (just as in inductive learning)
- Misconception 2: EBL is *competing* with inductive learning
  - In cases where we have weak domain theories, EBL can be seen as a “feature selection” phase for the inductive learner
- Misconception 3: *Utility* problem is endemic to EBL
  - Search control learning of any sort can suffer from utility problem
    - E.g. Using inductive learning techniques to learn search contr

L2P – Search Control - EBL

- EBL has been most popular approach to L2P technique in the early 90's
  - DOLPHIN, SNLP+EBL etc.
- Utility Problem
  - Too many learned rules can plague the planning system. Many of them are useless and it takes too much time to evaluate them during search (Minton)
- Potential Future Approach
  - Combine with MDL (Minimal Description Length) paradigm
  - Use EBL paradigm as feature selection approach
    - Note that Proof structure itself can be very useless, since only leaf node of the proof tree can be used as features
  - Simplify hypothesis space
    - Generally, ILP approaches did not work too well
    - Find alternative compact and modular KR (description logic?)

Proof Tree



Leaf node can directly be used for learning

L2P – Search Control – SAT constraints (controls SAT search, unit propagation)

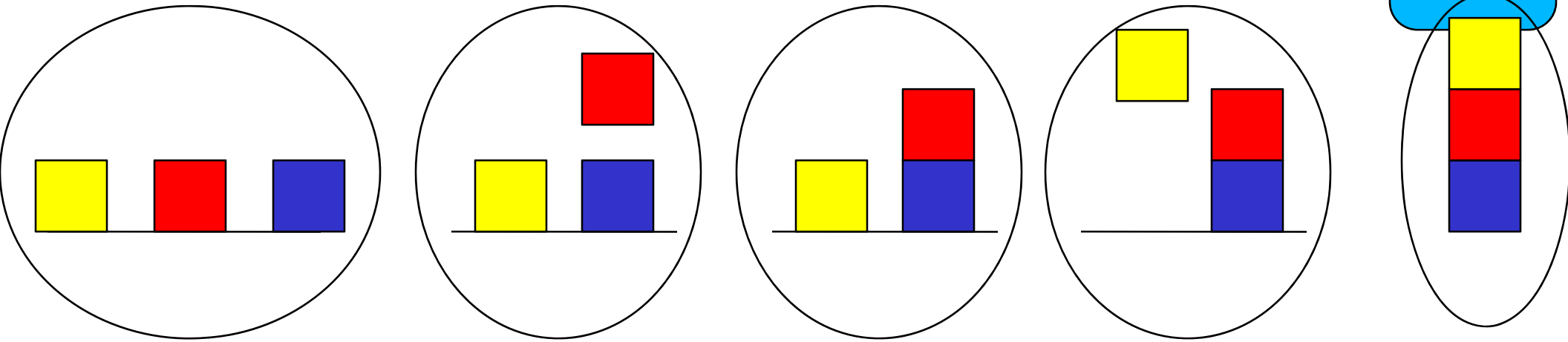
- The performance of SAT planner can be enhanced with domain background knowledge
  - For logistics domain,
    - Packages that are already in the goal shouldn't be moved
    - Once a package leaves a location, it should not return to it
    - A package can only be in original location or goal location
- Learning System
  - Huang, Selman and Kautz, 2000, ICML,
  - Generate training examples from solved plans

	Select Rule		Reject Rule	
	Positive Example	Negative Example	Positive Example	Negative Example
Static	Real	virtual	Virtual	Real
Dynamic	Real	Mutex Virtual	Mutex Virtual	Real

Introduction – What to watch for

- How the training data is obtained?
  - Self exploration or exercise?
  - From search tree?
  - User provided?
  - Automated planning results?
- How features are generated?
- What kind of background knowledge is needed?
  - We should be positive about background knowledge

L2P – Search Control – SAT constraints



Static/Dynamic
Select Positive

Static/Dynamic
Select Negative

Pickup Red

Pickup Yellow
Pickup Blue

Stack Red Blue

Putdown Red

Pickup Yellow

Unstack Red Blue

Stack Yellow Red

Putdown Yellow

L2P – Search Control – SAT constraints

- With positive and negative training examples, run FOIL to learn “selection” rules and “rejection” rules
- Use the learned rules to generate clauses for SAT
  - (pickup ?x) <- (clear ?x)
  - Generate (not (clear a)$_i$ V (pickup a)$_i$)
  - For ground facts and actions at levels I
- Experiments showed performance enhancement
- Future Approaches
  - Apply the learning to IP, LP, or CSP approaches
  - How to use stochastic rules, since learning can be imperfect
    - Maxsat

Approaches for Learning Search Control

"speedup learning"

Improve an existing planner

Learn "from scratch" how to plan

--Learn "reactive policies"
State x Goal→action

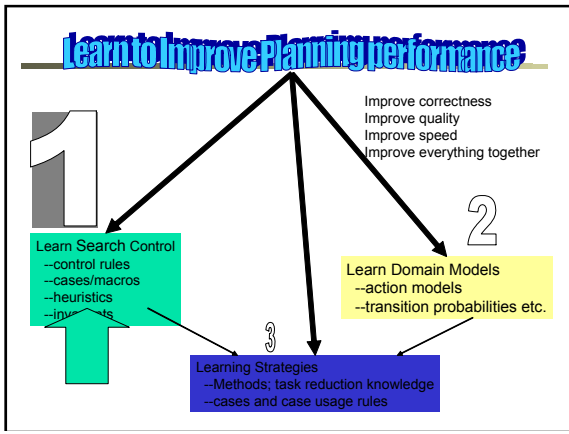
[Work by Khadron, 99;
Givan, Fern, Yoon, 2003→]

Learn rules
to guide choice points

Learn plans
to reuse

--Macros
--Annotated cases

Learn adjustments to
heuristics



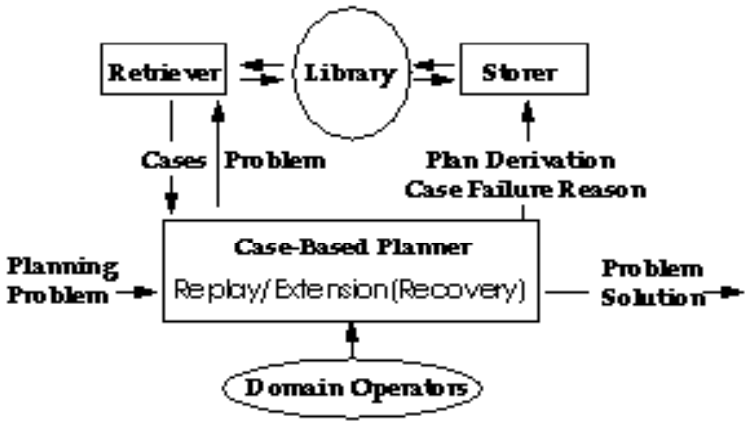
Case-based Planning Macrops, Reuse, Replay

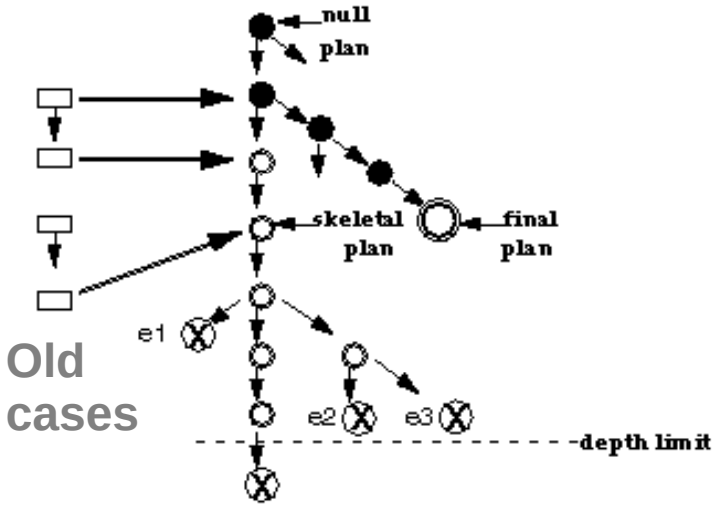
- Structures being reused
  - Opaque vs. Modifiable
  - Solution vs. Solving process (derivation/search trace)
- Acquisition of structures to be reused
  - Human given vs. Automatically acquired
    - Adapting human-given cases is a way of handling partial domain models
- Mechanics of reuse
  - Phased vs. simultaneous
- Costs
  - Storage & Retrieval costs; Solution quality

(Ihrig & Kambhampati, JAIR 97)

Case-study: DerSNLP

- Modifiable derivational traces are reused
- Traces are automatically acquired during problem solving
  - Analyze the interactions among the parts of a plan, and store plans for *non-interacting* subgoals separately
    - Reduces retrieval cost
  - Use of EBL failure analysis to detect interactions
- All relevant trace fragments are retrieved and replayed before the control is given to from-scratch planner
  - Extension failures are traced to individual replayed traces, and their storage indices are modified appropriately
    - Improves retrieval accuracy





L2P – Search Control - Macro

- From PDDL , for two actions, when effect of one is well connected to the precondition of the other, we can construct a macro action.
  - This can be verified from example solutions
  - A Macro is used as just an action during planning
  - Example, Push-Start and Push-End actions in Pipesworld domain (IPC-4)
- Similar to HTN structure learning. A learner can find frequent pattern in the solution plans
- Learning systems
  - MacroFF and Marvin
- Future Approaches
  - How to find longer Macros
  - Learn Macros from tagged solution trajectories

L2P – Search Control - MacroFF

```
(:action UNLOAD
:parameters
(?x - hoist ?y - crate ?t - truck ?p - place)
:precondition
(and (in ?y ?t) (available ?x) (at ?t ?p) (at ?x ?p))
:effect
(and (not (in ?y ?t)) (not (available ?x)) (lifting ?x ?y))
)
```

```
(:action DROP
:parameters
(?x - hoist ?y - crate ?s - surface ?p - place)
:precondition
(and (lifting ?x ?y) (clear ?s) (at ?s ?p) (at ?x ?p))
:effect
(and (available ?x) (not (lifting ?x ?y)) (at ?y ?p)
(not (clear ?s)) (clear ?y) (on ?y ?s))
)
```

```
(:action UNLOAD|DROP
:parameters
(?h - hoist ?c - crate ?t - truck ?p - place ?s - surface)
:precondition
(and (at ?h ?p) (in ?c ?t) (available ?h)
(at ?t ?p) (clear ?s) (at ?s ?p))
:effect
(and (not (in ?c ?t)) (not (clear ?s))
(at ?c ?p) (clear ?c) (on ?c ?s))
)
```

Reuse/Macrops Current Status

- Since ~1996 there has been little work on reuse and macrop based improvement of base-planners
  - People sort of assumed that the planners are already so fast, they can't probably be improved further
- Macro-FF, a system that learns 2-step macros in the context of FF, posted a respectable performance at IPC 2004 (but NOT in the KB-track)
    - Uses a sophisticated method assessing utility of the learned macrops (& also benefits from the FF enforced hill-climbing search)
      - Macrops are retained only if they improve performance significantly on a suite of problems
  - Given that there are several theoretical advantages to reuse and replay compared to Macrops, it would certainly be worth seeing how they fare at IPC [Open]

Approaches for Learning Search Control

"speedup learning"

Improve an existing planner

Learn "from scratch" how to plan

--Learn "reactive policies"
State x Goal→action

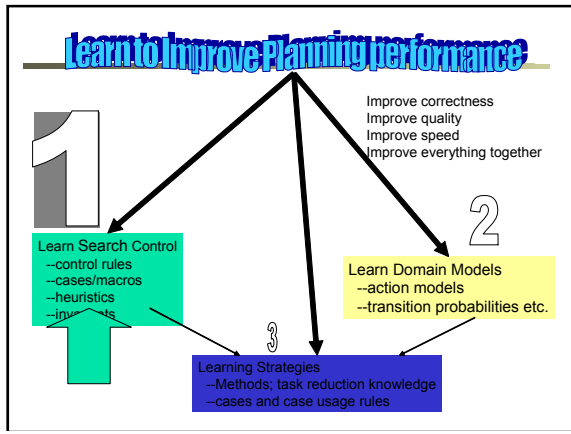
[Work by Khadron, 99;
Givan, Fern, Yoon, 2003→]

Learn rules
to guide choice points

Learn plans
to reuse

--Macros
--Annotated cases

Learn adjustments to
heuristics



Learning to Improve Heuristics

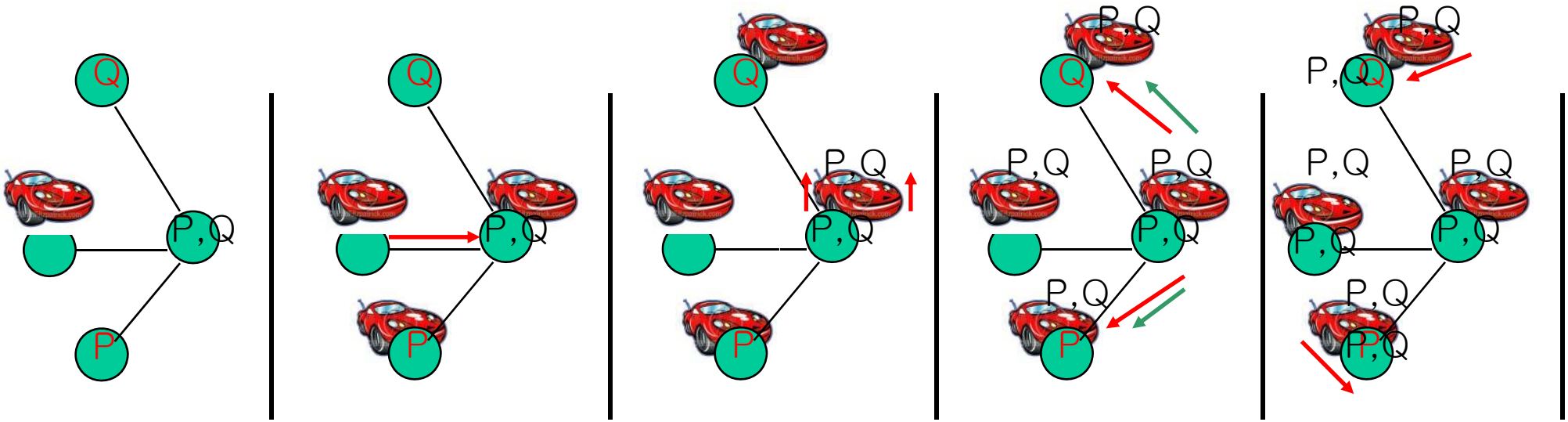
- Most modern planners use reachability heuristics
  - These approximate the reachability by ignoring some types of interactions (usually, negative interactions between subgoals)
  - While effective in general, ignoring such negative interactions can worsen the heuristic guidance and lead the planners astray
    - A way out is to "adjust" the reachability information with the information about interactions that were ignored
  1. (Static) Adjusted Sum heuristics as popularized in AltAlt
     - Increases the heuristic cost (as we need to propagate negative interactions)
     - Could be bad for progression planners which grow the planning graph once for each node..
  2. (Learn dynamically) Learn to predict the difference between the heuristic estimate by the relaxed plan and the "true" distance
     - Yoon et al. show that this is feasible—and manage to improve the performance of FF

[Yoon,Fern and Givan, ICAPS 2006]

L2P – Search Control – Heuristic Learning

- Relaxed plan based heuristic is the most successful and popular
- Sometimes, the heuristic estimate is drastically lower than the actual distance
- From solution trajectories, learn the difference between real distance and the heuristic estimate.
- With the learned difference function, we can adjust the heuristic estimation
- Important idea is how we draw features

Heuristic Value Comparison



Plangraph Length	4
Relaxed Plan Length (RPL)	7
Complementary Heuristic	1
Real Plan Length	8

Consider deletions of in(CAR,x) when move(x,y) is taken in relaxed plan

Learning Adjustments to Heuristics

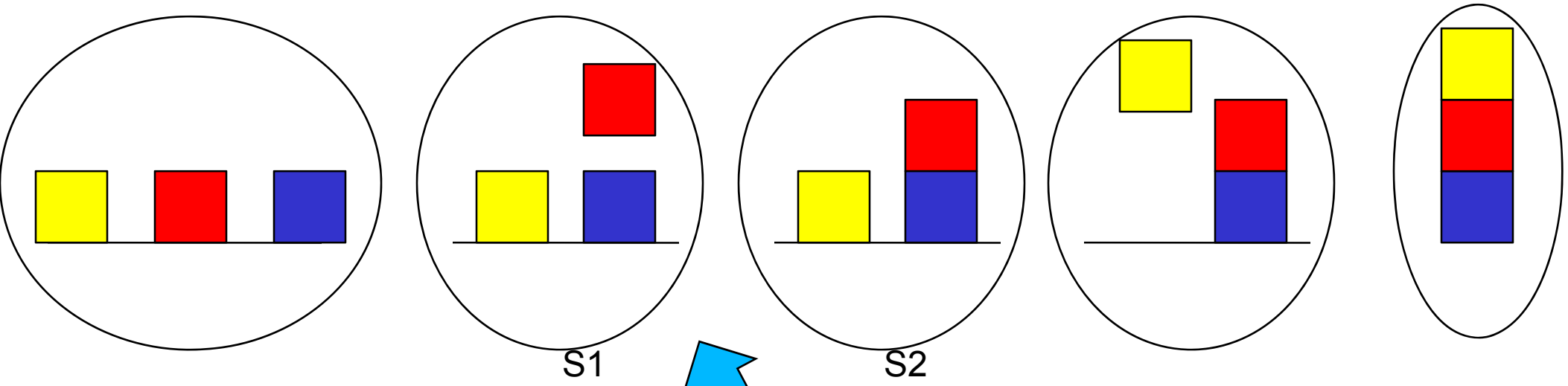
- Start with a set of training examples [Problem, Plan]
  - Use a standard planner, such as FF to generate these
- For each example [(I,G), Plan]
  - For each state S on the plan
    - Compute the relaxed plan heuristic S^R
    - Measure the actual distance of S from goal S* (easy since we have the current plan—assuming it is optimal)
  - Inductive learning problem
    - Training examples: Features of the relaxed plan of S
      - Yoon et al use a taxonomic feature representation Class labels: S^*-S^R (adjustment)
    - Learn the classifier
      - Finite linear combination of features of the relaxed plan

[Yoon et al, ICAPS 2006]

L2P – Search Control – Heuristic Learning

- The idea of using plangraph as a feature space is related to EBL
  - Plangraph is partial explanation of the potential plan
  - The learning finds flaw in the plangraph (or explantion) from training examples
  - Thus, this approach is a good example approach to EBL using weak or incomplete domain theory
    - Here relaxed operators are incomplete domain theory

L2P – Search Control – Beam Search



If the heuristic function is $\sum w_i * f_i$
We can update the weights w_i.
Those weights with features that are true in S2 will get higher while others get lower.

Similar to perceptron update

Xu, Fern and Yoon [IJCAI, 2007] show successful results on planning domains

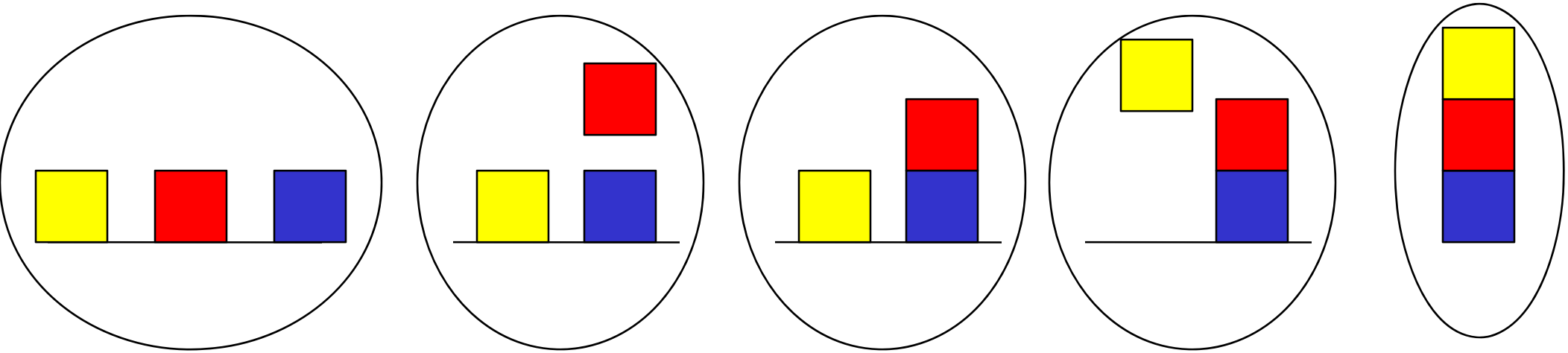
Consider a beam search and when we expand S1, we want to have S2 in the beam.
The beam maintains top 2 states in terms of heuristic value.

Suppose, S2 failed to get in the queue. Then we can update the heuristic function to include S2

L2P – Search Control – Measures Of Progress

- Measure of Progress in planning is some measure that monotonically increases (or decreases) with good plans – Parmar AAAI 02
- Example
  - The number of blocks in the good tower
  - The number of packages in the goal location
- The planning can be easier if we know such measure
  - We can safely use Enforced Hill Climbing approach
- The questions is how we automatically find such measure

L2P – Search Control – Measures Of Progress



- Given solution trajectories, a training example is consecutive two states in the trajectories, let the set of such states be J
- Find a measure I that increases most in J
- Add I to the tail of the measure list L
- Remove pair of states that are covered by I from J, set the new J and go back until J is empty

- Again the trick is using KR that is well suited for planning

What types of control knowledge used by KBplanners can be learned currently?

- Learnable (now)
  - Invariants
  - Simple search control rules
  - Adjustment rules for heuristics
- Open problems
  - Complex state sequence rules used by TLPlan
    - Invariants, a special case, is learnable
  - HTN schemas (or SHOP programs)
    - Recently, actively studied by researchers.
    - Ilghami et al, ICAPS ’06
    - Langley and Choi, JMLR ‘06

L2P – Search Control – How to Use

- Machine generated policies may not be complete
- Is there any way to intelligently leverage machine learned policies?
  - Discrepancy Search, follow the policy most of the times except for some limited amount of times
- Use discrepancy search in heuristic search produced better results
  - When a node A is being expanded, learned policy is applied to the node
  - Add all the nodes that occur along the policy from A, to the search queue
  - This is different from YAHSP or MacroFF style search
  - The intension is finding flaw of the input policy and fix it
  - Yoon, Fern and Givan, IJCAI 07, reported successful experimental results

Approaches for Learning Search Control

"speedup learning"

Improve an existing planner

Learn "from scratch" how to plan

--Learn "reactive policies"
State x Goal→action

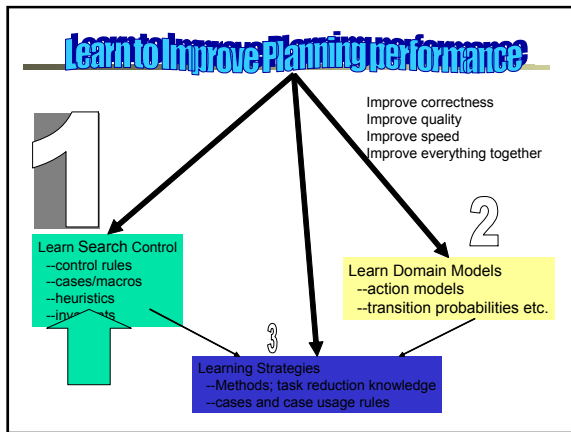
[Work by Khadron, 99;
Winner & Veloso, 2002;
Fern, Yoon and Givan, 2003→
Gretton & Thiebaux, 2004]

Learn rules
to guide choice points

Learn plans
to reuse

--Macros
--Annotated cases

Learn adjustments to
heuristics

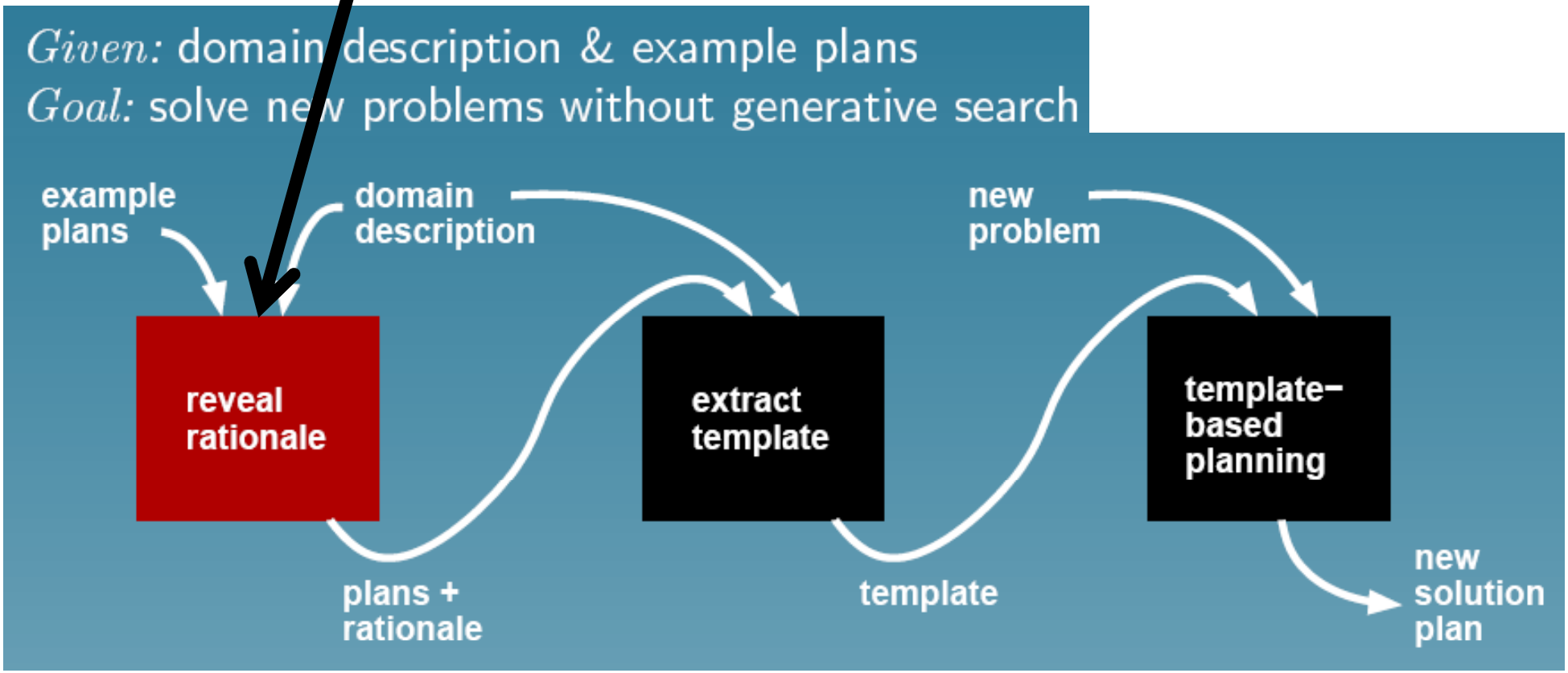


Learning From Scratch: Learning Reactive Policies

- We are not interested in speeding up an existing planner, but rather directly learn how to find plans for a large distribution of problems
- Specifically, we want to learn a policy of the form [state, goals]→action
  - Notice that this is a policy predicated both on states and goals (so it is supposed to work for all problems)
- Examples:
  - Winner & Veloso, 2002 (EBL/Induction)
  - Khadron, 99 (Also, Martin/Geffner; 2002) (Induction)
  - Fern, Yoon and Givan, 2003 (also Gretton & Thiebaux, 2004) (Induction; Reinforcement Learning)

Learning Domain Specific Planners

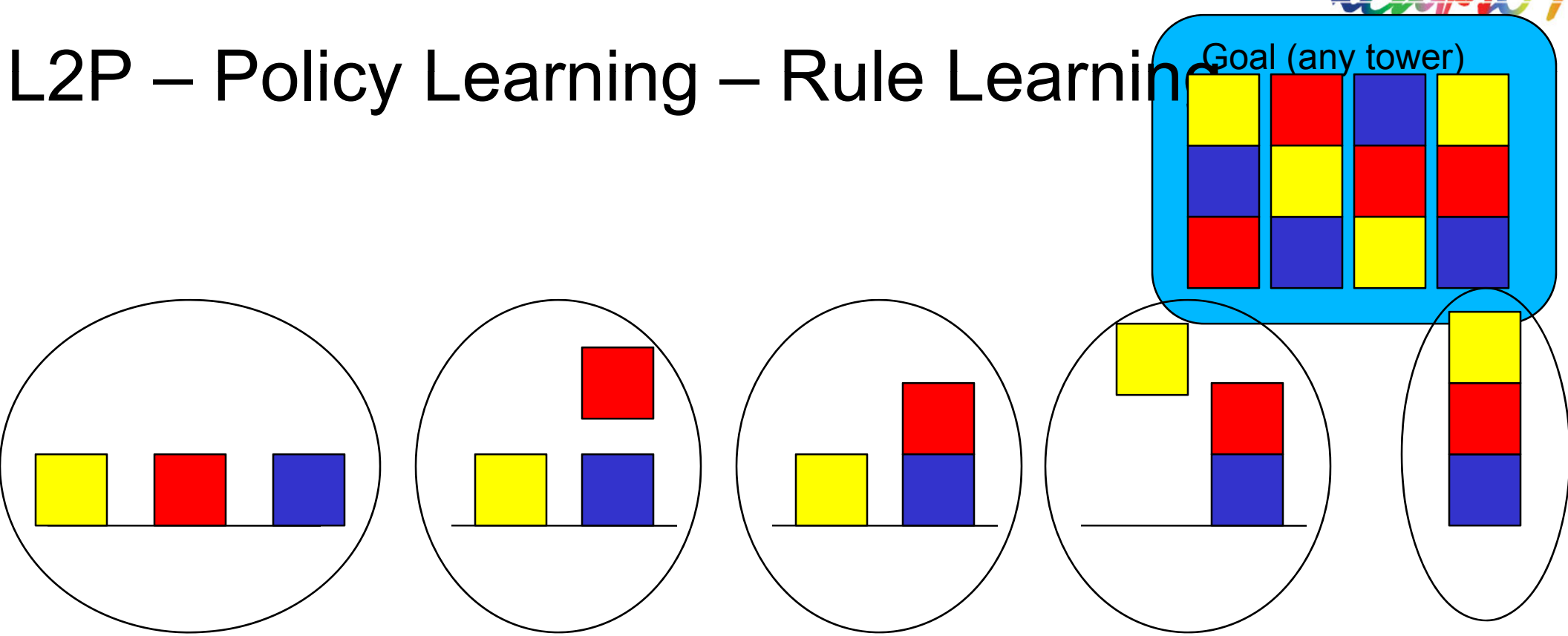
Explanation-based Generalization!



L2P – Policy Learning – Rule Learning

- Khardon MLJ '99 provided theoretical proof [L2ACT]
  - If one can find a deterministic strategy that can find trajectories close to the training trajectories, then the strategy can perform as good as the provider of the training trajectories
  - One can view the strategy as a deterministic policy in MDP
    - A policy is a mapping from state to action
- Martin and Geffner, KR 2000, developed a policy learning system for Blocksworld
  - Showed the importance of KR
  - Used Description Logic to compactly represent "good tower" concept
- Yoon, Fern and Givan, UAI '02, developed a policy learning system for first order MDPs
- These systems use decision-rule representations

L2P – Policy Learning – Rule Learning



Positive Example

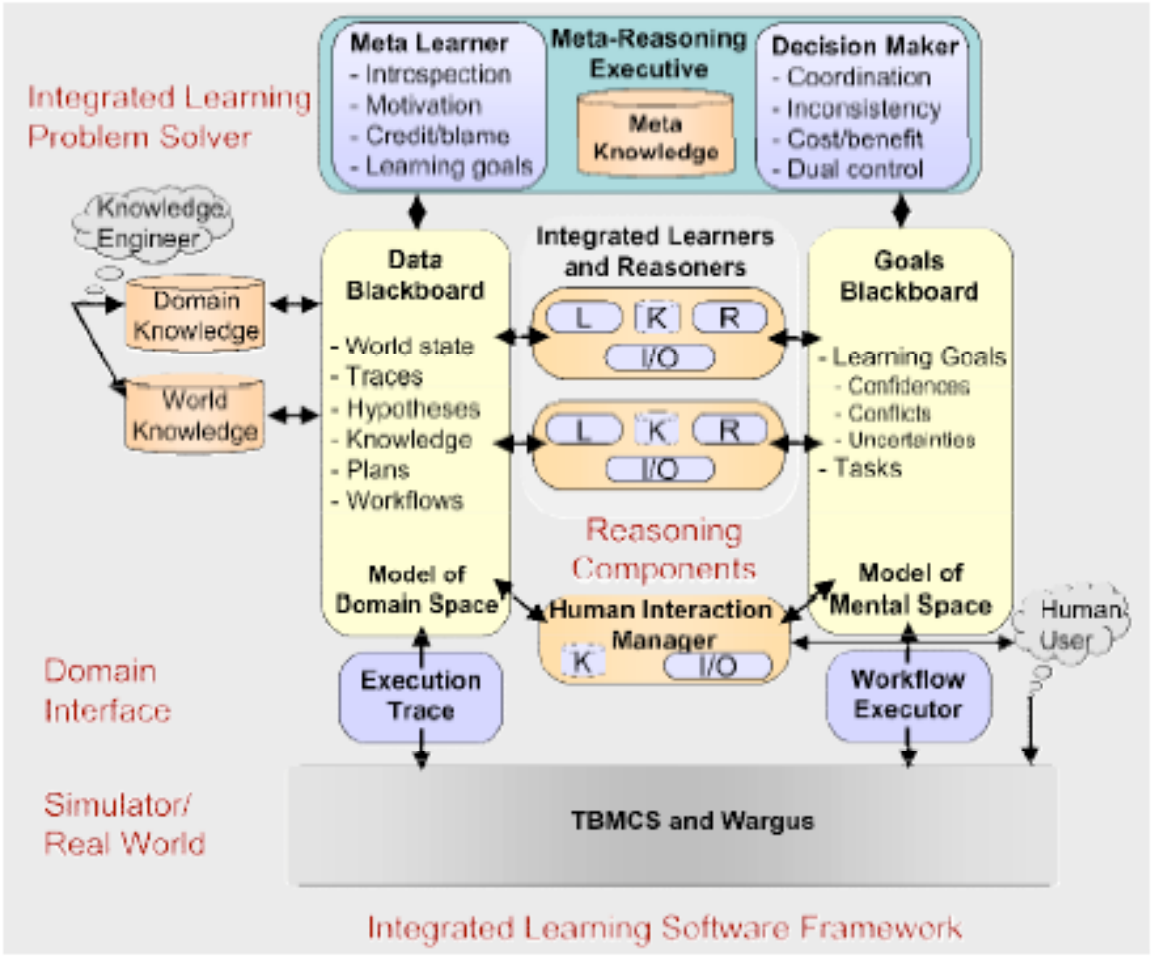
Negative Example

- Pickup Red
- Pickup Yellow
- Pickup Blue
- Stack Red Blue
- Putdown Red
- Pickup Yellow
- Unstack Red Blue
- Stack Yellow Red
- Putdown Yellow

Though Pickup Red was selected in the first state, other actions are equally good

L2P – Policy Learning – Rule Learning

- Treat each state-action pair separately from solution trajectories, let the pairs be J
  - Try to find the action-selection-rule l that covers J most well
  - Add l to the tail of L, the decision list
  - Remove state-action pairs, covered by the rule l. Set the new J and go back until there is no remaining state-action pair
- Rule learning technique is one of the most successful learning techniques for planning in modern era
- Martin and Geffner/Yoon,Fern and Givan showed the importance of KR
- The learning technique can be applied to any reactive style control, so can be applied to POMDP, Stochastic Planning as well as Conformant or Temporal Planning
- Future Approaches
  - Develop KR that suits well to the Conformant or Temporal planning and apply the learning technique



- **Goal-driven, explanation-based learning approach of GILA gleans and exploits knowledge in multiple natural formats**
  - Single example analyzed under the lenses of multiple "ILRs" to learn/improve
    - Planning operators
    - Task networks
    - Planning cases
    - Domain uncertainty

Challenges and Solutions

- Domain Knowledge in multiple modalities
  - → Use multiple ILRs customized to different types of knowledge
- Learning in multiple time-scales
  - → Combine eager (e.g. EBL-style) and lazy (e.g. CBR-style) learning techniques
- Handling partially correct domain models and explanation
  - → Use local closed world assumptions
- Avoiding balkanization of learned knowledge
  - → Use structured explanations as a unifying "glue"
- Meeting explicit learning goals
  - → Use Goal-driven meta-learning techniques

Gila - Performance Review

Policy Learner

Scenario	DTL	SPL	CBL	QOS Average	QOS Median
1	39%	47%	8%	61.4	70
2	49%	27%	19%	40.0	10
3	26%	52%	17%	55.6	70
4	23%	77%	0%	76.7	80
5	43%	51%	6%	65.0	70
6	32%	63%	0%	83.8	90

Numbers of DTL, SPL, and CBL : The percent of contribution of each ILR component. The percent of the actually used solution suggested by each component

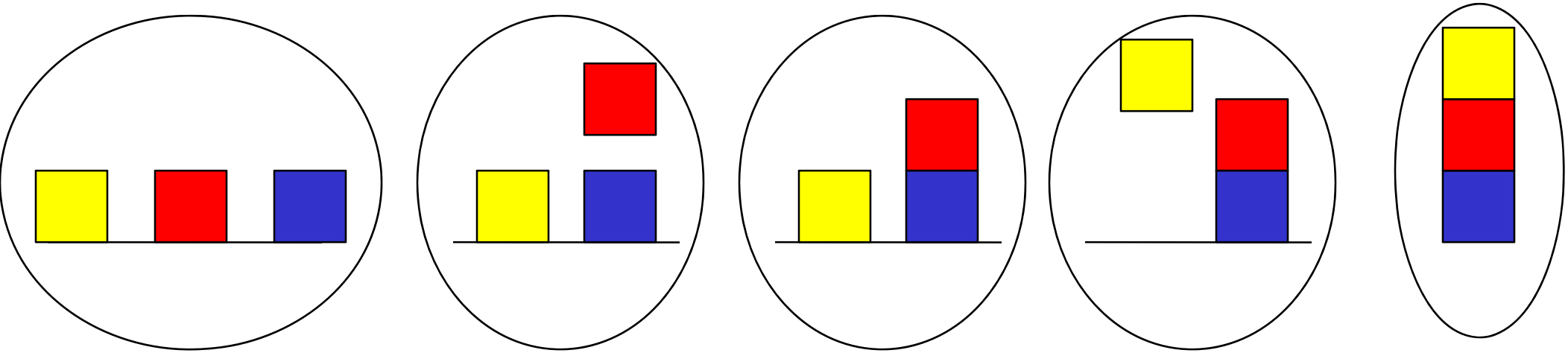
QOS means the quality of each Pstep

As SPL participates in the solution more, the QOS improves

L2P – Policy Learning- RRL

- [RRL] is a relational version of reinforcement learning – Dzeroski, De Raedt, and Driessens, MLJ, 2001
- Has been successfully applied to some versions of Blocksworld and games
- Used TILDE, relational tree regression technique to learn Q-value functions, which scores State-Action pair
- Later Direct Policy learning has been shown to be better than Q-value learning

L2P – Policy Learning - RRL



0.72 (Pickup Red)

(ontable Red)
(ontable Blue)
(ontable Yellow)

0.81: (stack Red Blue)

(holding Red)
(ontable Blue)
(ontable Yellow)
(clear Blue)

0.9: (pickup Yellow)

(on Red Blue)
(ontable Yellow)
(clear Red)
(clear Yellow)

1: (stack Yellow Red)

(holding Yellow)
(on Red Blue)
(clear Red)

L2P – Policy Learning - RRL

- Merit: doesn't have to worry about positive/negative examples from trajectories
- Model-free: RRL does not need domain model
- Does not need teacher.
- Slow-convergence: especially when it is very hard to find goals
  - We will deal with this in the next technique
- Future Approaches
  - Concept-language based reinforcement learning techniques

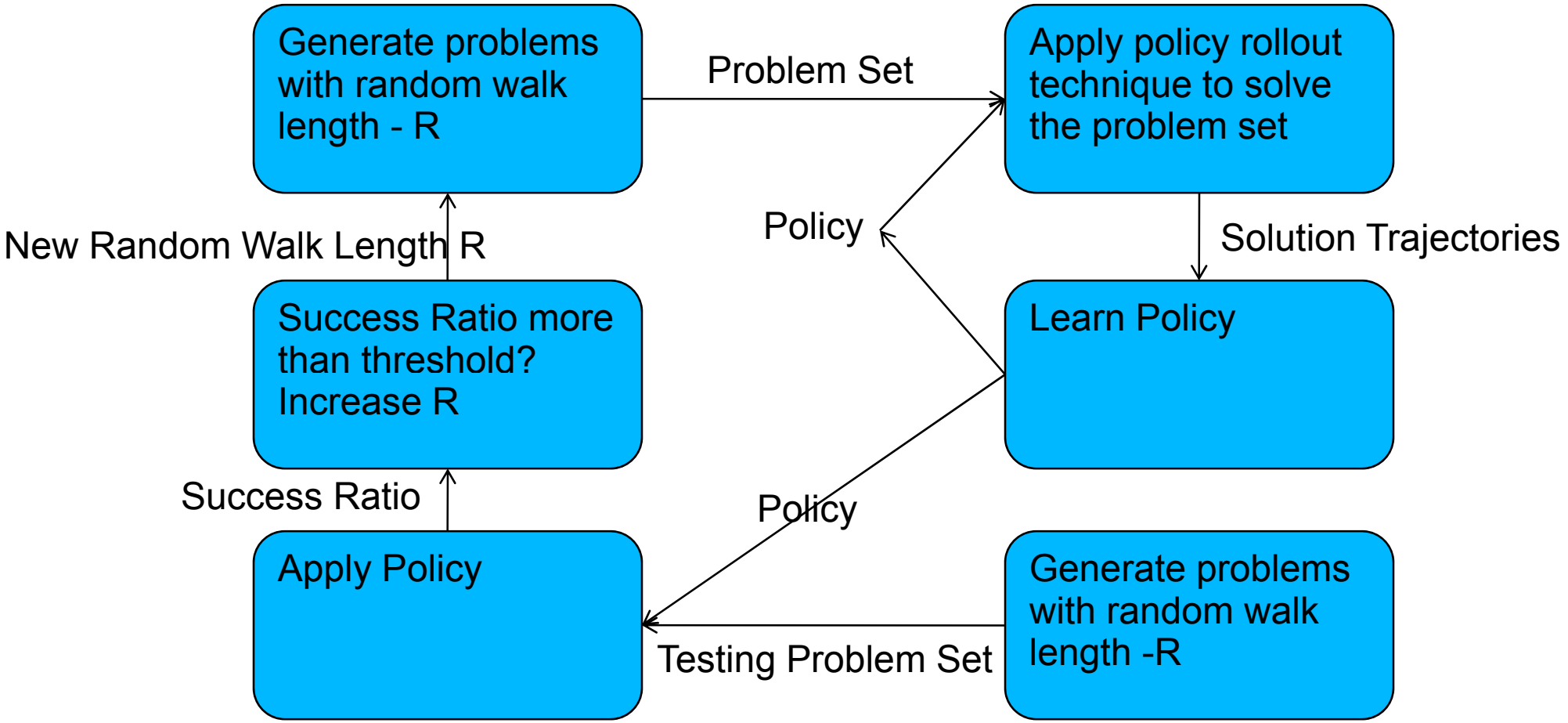
RL and MDP

- One of the Primitive approach to Planning and learning is Reinforcement Learning (RL).
  - Model-Free RL combines speed-up and domain learning aspect
  - Model-based RL achieves speed-up planning
- Solution techniques to Markov Decision Processes (MDP) problems are related to L2P
  - Finding policies
  - Learning Approximate Value Function
  - Learning Policy
- RL and MDP techniques does not scale well
  - All the state space needs to be enumerated
- We need scalable planning to deal with real world

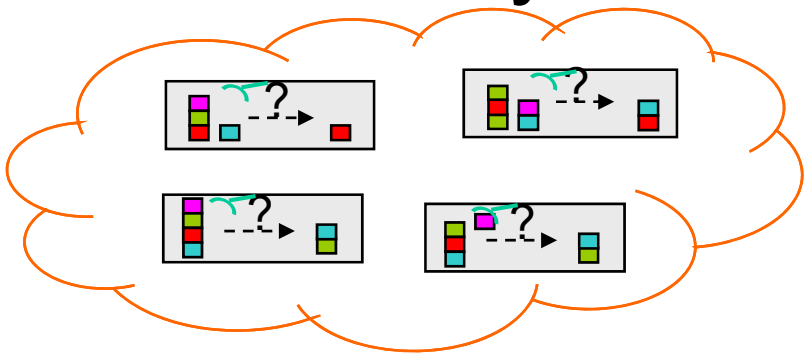
L2P – Policy Learning – Automated Domain Solver

- When we have a good policy learner but does not have a teacher who provides solution trajectories to the target domain
  - How we get training data?
- Fern,Yoon and Givan, JAIR '06, developed an interesting technique based on random walk idea
- Easy problems can easily be generated with small random walk
  - The end of the random walk is the goal state
  - One can imagine that as the random walk length increases the problems will become harder
  - It is not guaranteed that random-walk generated problem set is close to the real distribution of the planning problems
    - Consider FreeCell
    - However, in practice, this idea produced good results across benchmark domains
- Use approximate policy iteration technique
- The technique has been successfully applied to both deterministic and probabilistic planning domains

L2P – Policy Learning – Automated Domain Solver



Approximate Policy Iteration



Planning Domain
(problem distribution)

current policy π

?

trajectories of
improved policy π'

Learn approximation
of π'

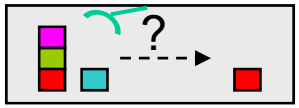
π'

Control Policy

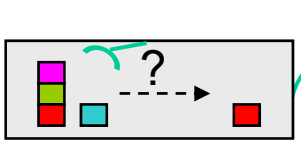
Introduction – What to watch for

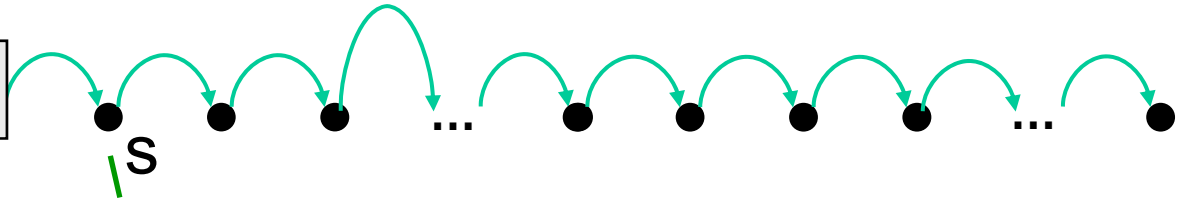
- How the training data is obtained?
  - Self exploration or exercise?
  - From search tree?
  - User provided?
  - Automated planning results?
- How features are generated?
- What kind of background knowledge is needed?
  - We should be positive about background knowledge

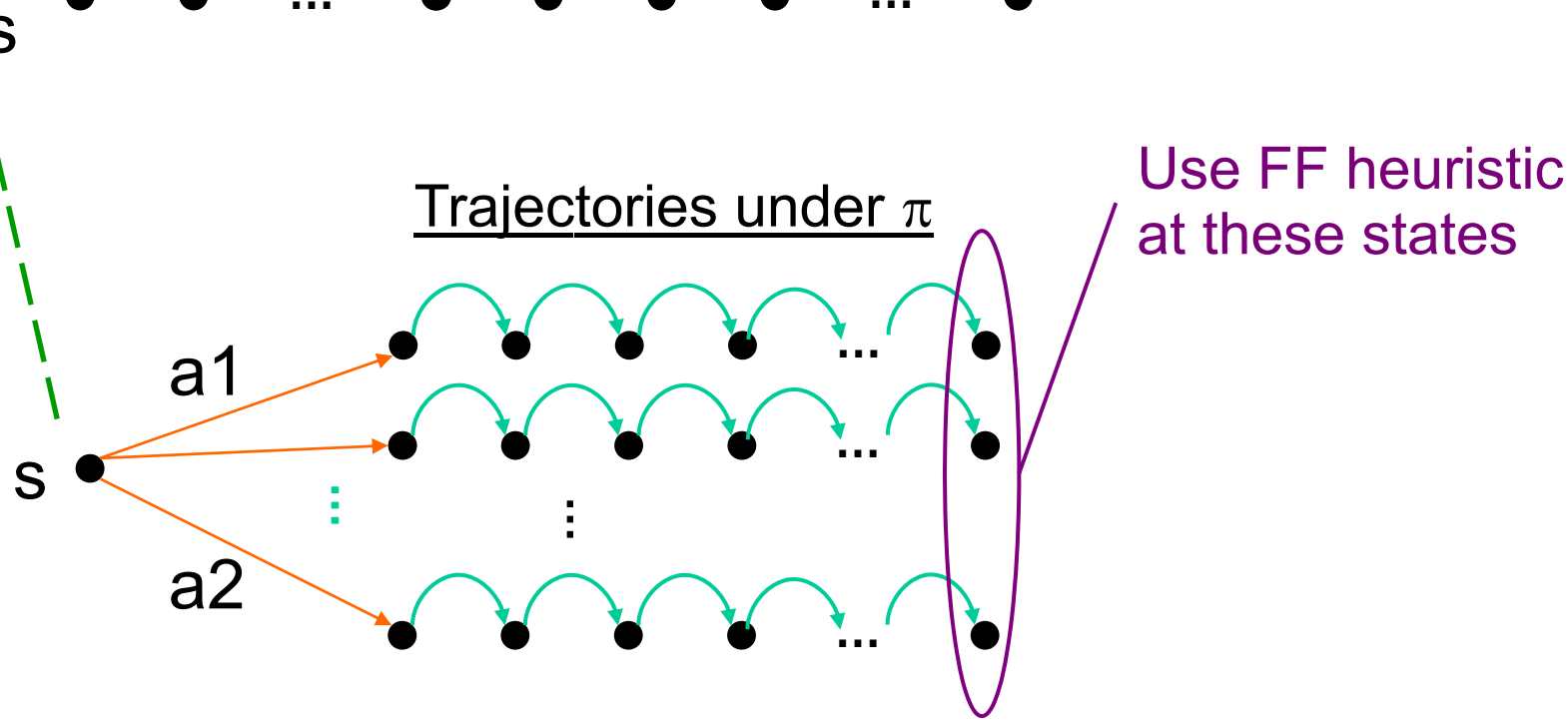
Computing π' Trajectories from π

Given: current policy π and problem 

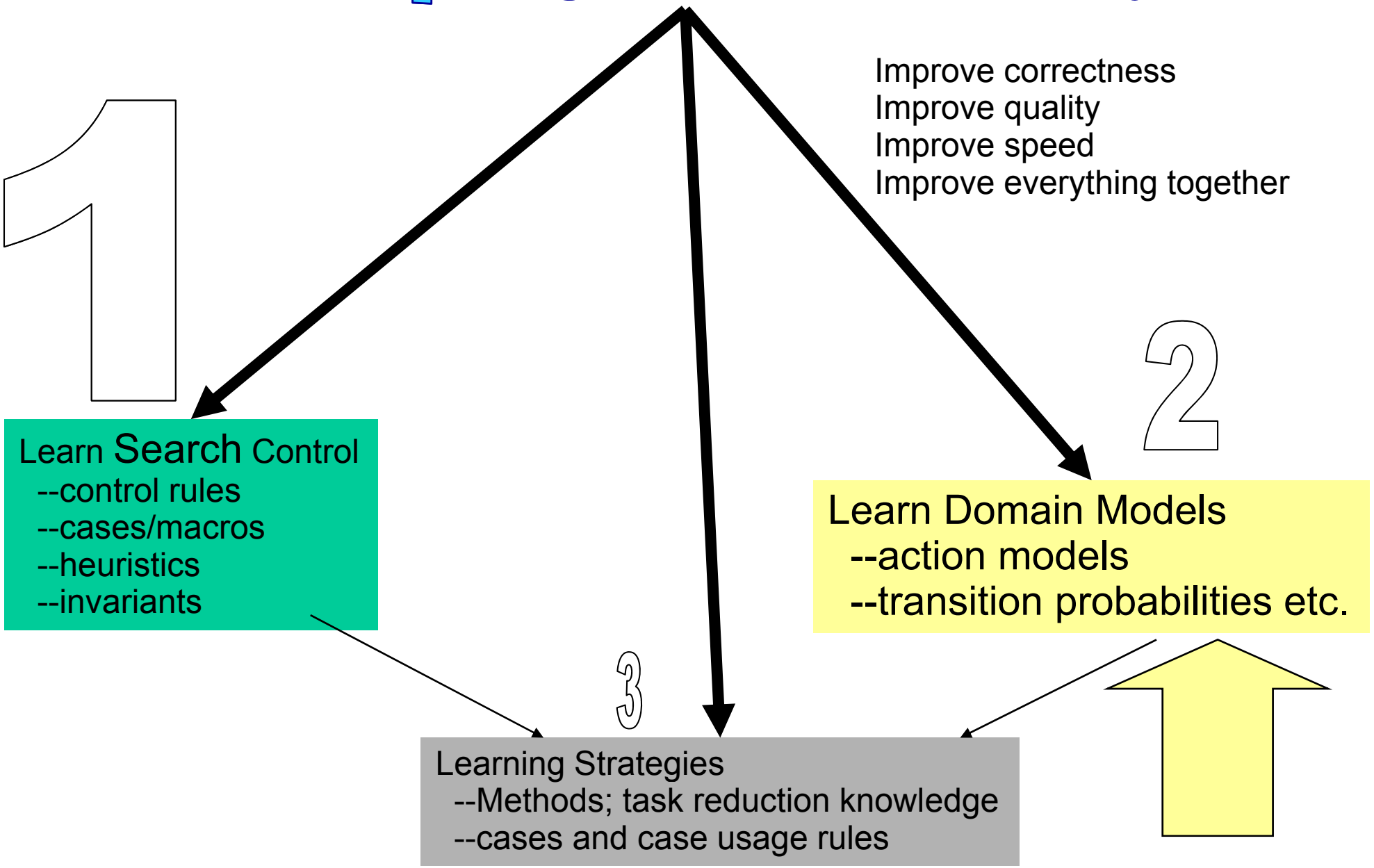
Output: a trajectory under improved policy π'



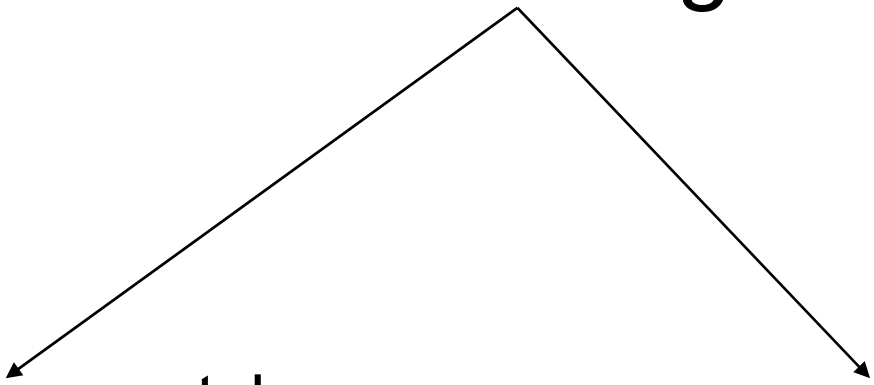




Learn to Improve Planning performance



Learning Domain Knowledge



Learning from scratch

→Operator Learning

Operationalizing existing knowledge

→EBL-based operationalization [Levine/DeJong; 2006]

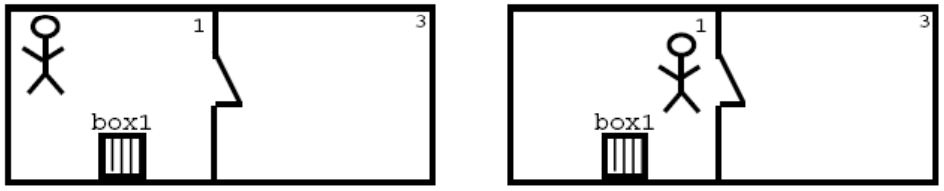
→RL for focusing on “interesting parts” of the model …lots of people including [Aberdeen et. Al. 06]

Learning Domain Knowledge (From observation)

- Learning Operators (Action Models)
  - Given a set of [Problem; Plan: (operator sequence)] examples; and the space of domain predicates (fluents)
  - Induce operator descriptions
    - Operators will have more parameters in expressive domains
      - Durations and time points; probabilities of outcomes etc.
- Dimensions of variation
  - Availability of intermediate states (Complete or Partial)
    - Makes the problem easy—since we can learn each action separately. Unrealistic (especially “complete” states)
  - Availability of partial action models
    - Makes the problem easier by biasing the hypotheses (we can partially explain the correctness of the plans). Reasonably realistic.
  - Interactive learning in the presence of humans
    - Makes it easy for the human in the loop to quickly steer the system from patently wrong models

OBSERVE: Assumes full knowledge of intermediate states

- When we have full knowledge of intermediate states, the action effects are easy to learn!
  - Preconditions need to be learned by inducing over a sample of states where the action is applicable
  - Zettlemeyer et al do this for stochastic case [2005]




```
(operator goto-dr
  (preconds ((<v1> door) (<v2> object) (<v3> room) (<v4> room))
   (and (inroom robot <v4>)
        (connects <v1> <v3> <v4>)
        (connects <v1> <v4> <v3>)
        (dr-to-rm <v1> <v3>)
        (dr-to-rm <v1> <v4>)
        (unlocked <v1>)
        (dr-closed <v1>)
        (arm-empty)
        (inroom <v2> <v3>)
        (pushable <v2>)))
  (effects nil ((add (next-to robot <v1>)))))
```

(connects dr12 rm2 rm1)	(dr-open dr12)	(connects dr12 rm2 rm1)	(dr-open dr12)
(connects dr12 rm1 rm2)	(unlocked dr12)	(connects dr12 rm1 rm2)	(unlocked dr12)
(dr-to-rm dr12 rm1)	(inroom robot rm1)	(dr-to-rm dr12 rm1)	(inroom robot rm1)
(dr-to-rm dr12 rm2)	(inroom c rm2)	(dr-to-rm dr12 rm2)	(inroom c rm2)
(pushable c)	(inroom b rm1)	(pushable c)	(inroom b rm1)
(pushable b)	(arm-empty)	(pushable b)	(arm-empty)
(inroom a rm1)	(next-to robot a)	(inroom a rm1)	(next-to robot dr12)
(a) Pre-state		*(b) Post-state*	

Delta-state = add: (next-to robot dr12) del: (next-to robot a)

[Wang, 1994]

L2P – Domain Learning – Logical Filtering

- Chang and Amir, ICAPS '05, applied logical filtering approach to learning domain transition models
- Maintain both Belief state and Domain Transition Models
- Update the belief state and domain transition models with logical filtering
- Thus a belief state in this work is a pair of belief state and transition model
- The approach has been successfully to propositional domains
- Logical filtering can be a good candidate for domain learning

L2P – Domain Learning – Learn Probabilistic Operators

- Zettlemoyer, Pasula and Kaelbling, AAAI, 2005, learned probabilistic planning operators from simulated blocksworld
  - Includes precondition and effects
  - Used deictic representation

pickup(X) : Y : on(X, Y), Z : table(Z) inhand-nil
:
.80 : ¬*on(X, Y), inhand(X),*¬*inhand-nil, clear(Y)*
.10 : ¬*on(X, Y), on(X, Z), clear(Y)*
.10 : no change
where *Y is now defined as a deictic*

L2P – Domain Learning – Learn Probabilistic Operators

LearnRuleSet(E)
Inputs:
Training examples E
Computation:
Initialize rule set R to contain only the default rule
While better rules sets are found
For each search operator O
Create new rule sets with O, RO = O(R,E)
For each rule set R0 in RO
If the score improves (S(R0) > S(R))
Update the new best rule set, R = R0
Output:
The final rule set R

The learned operators were tested by planning with the operators. With learned operators, the planner could perform well on the task of stacking blocks

There are 8 methods for the enumeration of the new rule set. One of them is EBL

The learning and planning system involves visual interpretation and rigid body models. Thus very close to real world environment

ARMS

(Doesn't assume intermediate states; but requires action parameters)

- Idea: See the example plans as "constraining" the hypothesis space of action models
  - The constraints can be modeled as SAT constraints (with variable weights)
  - Best hypotheses can be generated by solving the MAX-SAT instances
- Performance judged in terms of whether the learned action model can *explain* the correctness of the observed plans (in the test set)
  - Notice that if my theory is incomplete I might be able to explain both correct and incorrect plans to be correct…

- Constraints
  - Actions' preconditions and effects must share action parameters
  - Actions must have non-empty preconditions and effects; Actions cannot add back what they require; Actions cannot delete what they didn't ask for
  - For every pair of frequently co-occurring actions ai-aj, there must be some causal reason
    - E.g. ai must be giving something to aj OR ai is deleting something that aj gives

[Yang et. al. 2005]

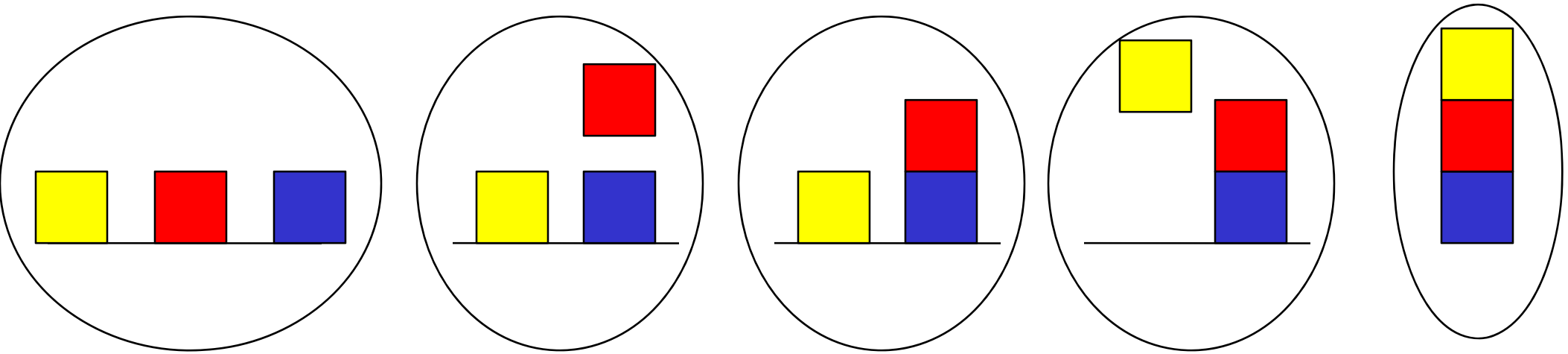
L2P – Domain Learning - MLN

- We can use existing Machine Learning package like (Markov Logic Network) MLN to learn domain operators
- Yoon and Kambhampati, ICAPS '07 workshop, showed learning and planning approaches based on MLN
- Learning
  - Separate precondition axiom and effect axiom
    - This has been used by Kautz and Selman
  - Update the axioms from observations using MLN tool
  - Action -> Precondition (in the current state)
  - Action -> Effect (next state)
- Planning
  - Construct probabilistic plangraph with learned axioms
  - View the plangraph as Bayes Net
    - Precondition and effect are conditional upon actions
    - Prior action probabilities are specified as .5
  - View initial state and goal state as evidence variables and solve for MPE

L2P – Domain Learning - MLN

- Operators can be represented with probabilistic logic or Markov Logic Network (MLN)
  - Precondition : Action -> Precondition (relation between current state and action in the state)
  - Effect: Action->Precondition (relation between the current action and the next state)
  - After training, axioms will have weight
    - Frequently verified axioms will have higher weight
    - None observed axioms will have lower weight
- If random wondering produced the trajectory, S_1,A_1, ……, S_n
  - (S_1,A_1), …., (S_{n-1},A_{n-1}) are training examples for precondition axiom
  - (A_1,S_2), ….., (A_{n-1},S_n) are training examples for effect axiom

L2P - Domain Learning - MLN



(armempty)
(ontable Y)
(ontable R)
(ontable B)
(clear R)
(clear B)
(clear Y)

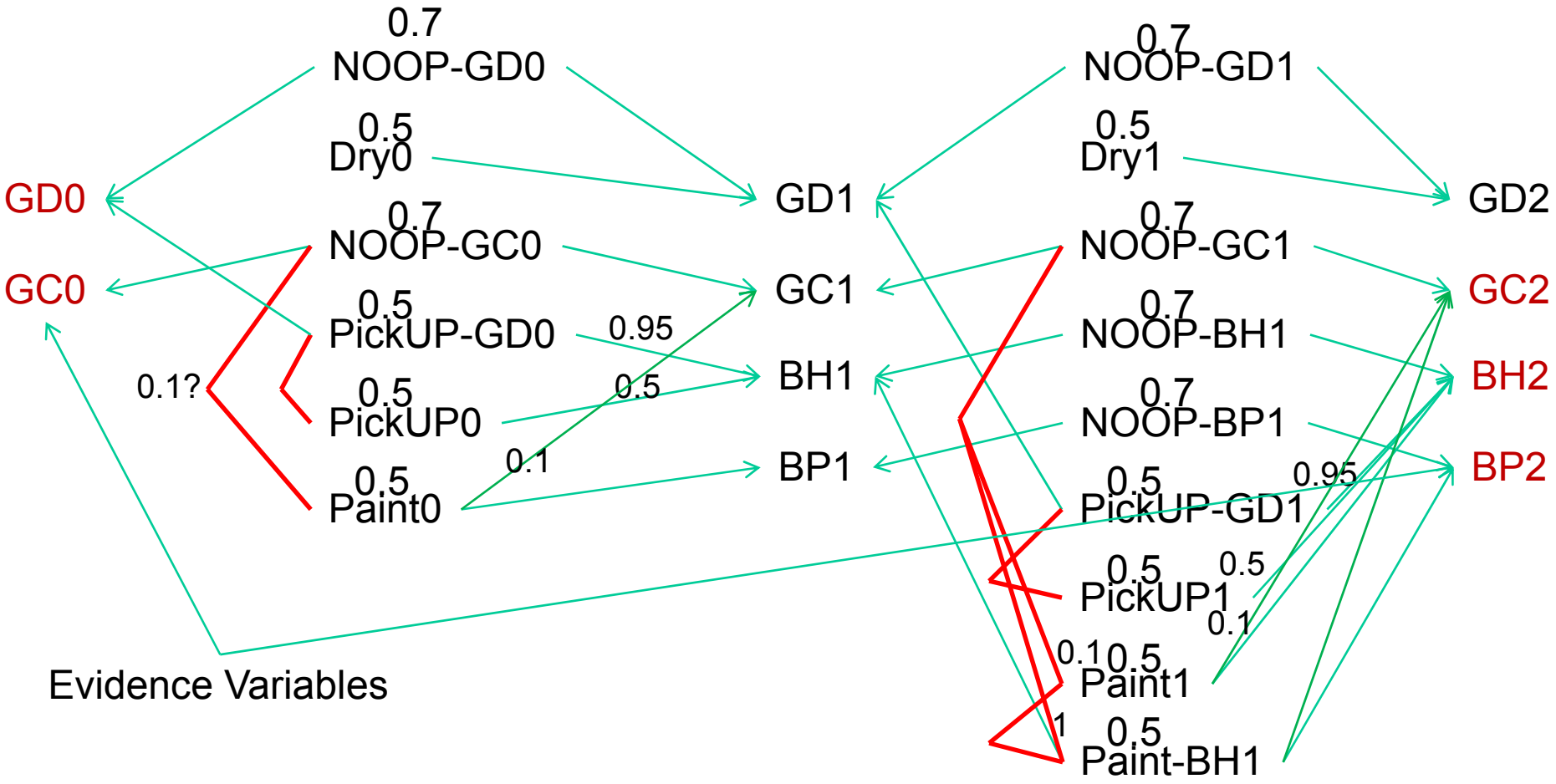
(holding R)
(clear Y)
(clear B)
(ontable Y)
(ontable B)

Precondition Axiom
(Pickup ?x) → (armempty), 0.5 → 0.7

Effect Axiom
(Pickup ?x) → NOT(armempty) 0.5 → 0.7

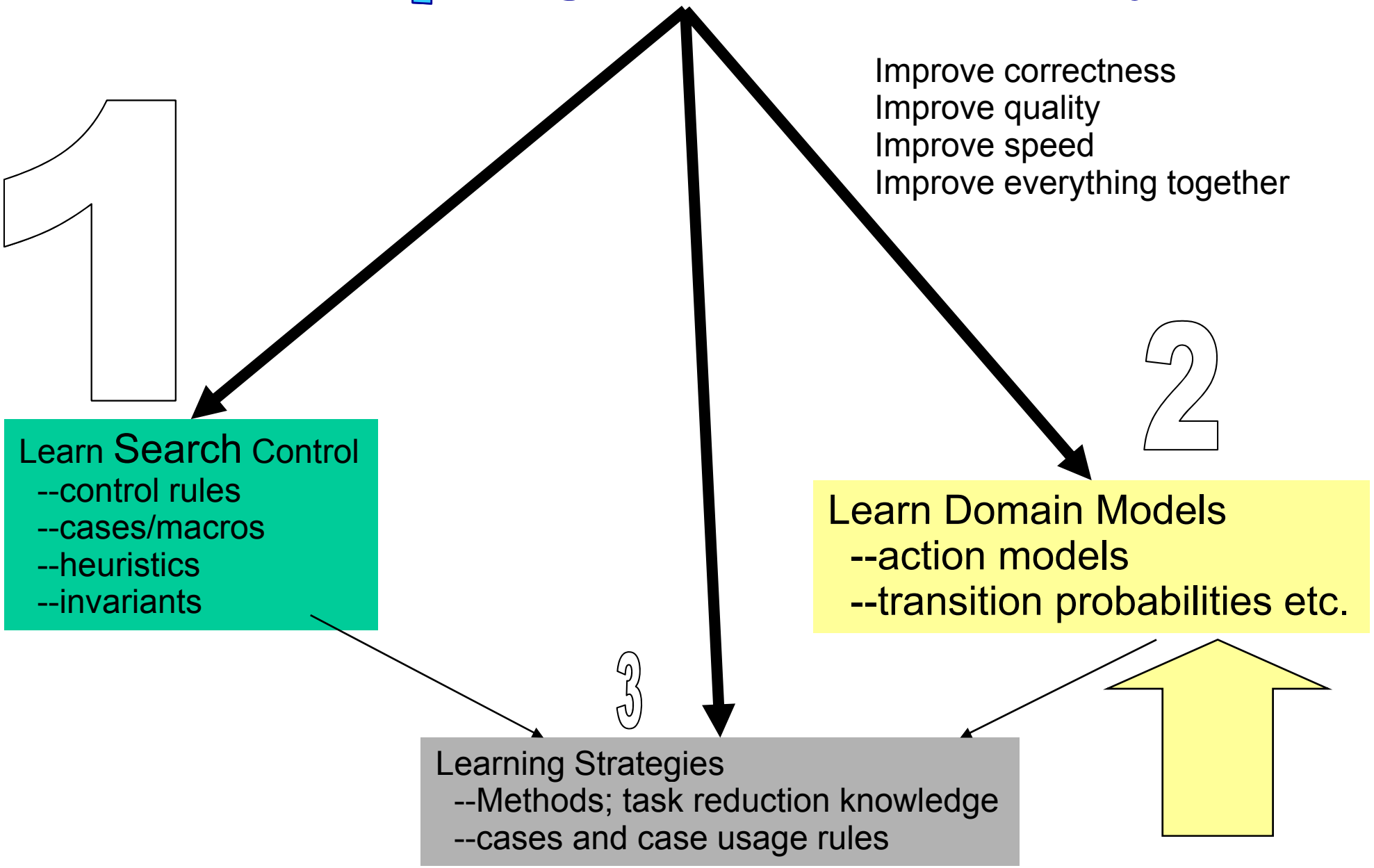
Pickup R

Plangraph as Bayes Net



How we find a solution?
MPE (most probabilistic explanation)
There are some solvers out there

Learn to Improve Planning performance



L2P - Challenges

- Machine Learned Knowledge is still not as good as human provided one
  - Consider TLPlan or Shop
- Most of the machine learning approaches to planning has been on classic planning
  - Temporal, or Oversubscribed planning
- Domain Learning
- How to plan, when domain is incomplete (since machine learning does not guarantee) and control knowledge is also incomplete
  - Model-lite Planning
- Use Partial Domain Knowledge for Learning
- Applying L2P techniques to ORTS

Summary

- Learning methods have been used in planning for both improving search and for learning domain physics
  - Most early work concentrated on search
  - Most recent work is concentrating on learning domain physics
    - Largely because we seem to have a very good handle on search
- Most effective learning methods for planning seem to be:
  - Knowledge based
    - Variants of Explanation-based learning have been very popular
  - Relational
- Many neat open problems...

References

- DerSNLP (Ihrig and Kambhampati, AAAI, 1994)
- [MLP] Model-lite Planning (Kambhampati, AAAI, 2007)
- [RL] Reinforcement Learning: A Survey (Kaelbling, Littman and Moore, JAIR, 1996)
- [NDP] Neuro-Dynamic Programming (Bertsekas and TsiTsiklis, Athena Scientific)
- Learning-Assisted Automated Planning: Looking Back, Taking Stock, Going Forward (Zimmerman and Kambhampati, AI Magazine, 2003)
- STRIPS (Fikes and Nilsson, 1971)
- [HAMLET] Lazy incremental learning of control knowledge for efficiently obtaining quality plans. AI Review Journal. Special Issue on Lazy Learning, (Borrajo and Veloso) February 1997
- Learning by experimentation: The operator refinement method. (Carbonell and Gil) Machine Learning: An Artificial Intelligence Approach, Volume III, 1990.
- [RRL] Relational reinforcement learning. Machine Learning, (Dzeroski, De Raedt and Driessens) 2001.
- Learning to improve both efficiency and quality of planning. (Estlin and Mooney) IJCAI, 1997
- [TIM] The automatic inference of state invariants in tim. (Fox and Long), JAIR, 1998.
- [DISCOPLAN] Discovering state constraints in DISCOPLAN: Some new results. (Gerevini and Schubert), AAAI 2000

References

- [Camel] Camel: Learning method preconditions for HTN planning (Ilghami, Nau, Munoz-Aila and Aha) AIPS, 2002
- [SNLP+EBL] Learning explanation-based search control rules for partial order planning. (Katukam and Kambhampati), AAAI, 1994
- [L2ACT] Learning action strategies for planning domains. (Khardon) Artificial Intelligence, 1999.
- [ALPINE] Learning abstraction hierarchies for problem solving. (Knoblock), AAAI, 1990
- [SOAR] Chunking in SOAR: The anatomy of a general learning mechanism. (Laird, Rosenbloom and Newell) 1986.
- Machine Learning Methods for Planning. (Minton and Zweben) Morgan Kaufmann, 1993.
- [DOLPHIN] Combining FOIL and EBG to speed-up logic programs. (Zelle and Mooney)IJCAI 1993.
- [TLPlan] Using Temporal Logics to Express Search Control Knowledge for Planning, (Bacchus and Kabanza), AI, 2000
- [PDDL] The Planning Domain Definition Language, (McDermott),
- [Graphplan] Fast Planning Through Planning Graph Analysis (Blum and Furst), AI, 1997
- [FF] The FF Planning System: Fast Plan Generation Through Heuristic Search, (Hoffmann and Nebel) JAIR, 2001
- [Satplan] Planning as Satisfiability, (Kautz and Selman), ECAI, 1992
- [IPPlan] On the use of integer programming Models in AI Planning, (Vossen, Ball, Lotem and Nau), IJCAI, 1999
- [SGPlan] Hsu, Wah, Huang and Chen

References

- [Yahsp] , A Lookahead Strategy for Heuristic Search Planning, (Vidal), ICAPS 2004
- [Macro-FF] , Improving AI planning with automatically learned macro operators (Botea, Enzenberger, Muller, and Schaeffer), JAIR, 2005
- [Marvin] , Online Identification of Useful Macro-Actions for Planning, (Coles and Smith), ICAPS, 2007
- Learning Declarative Control Rules for Constraint-Based Planning, (Huang, Selman and Kautz), ICML, 2000
- [FOIL], FOIL: A Midterm Report, (Quinlan and Cameron-Jones), ECML, 1993
- [Martin and Geffner] Learning Generalized Policies in Planning Using Concept Languages, KR, 2000
- Inductive Policy Selection for First-Order MDPs, (Yoon, Fern, Givan), UAI, 2002
- Learning Measures of Progress for Planning Domains, (Yoon, Fern and Givan), AAAI, 2005
- Approximate Policy Iteration with a Policy Language Bias: Learning to Solve Relational Markov Decision Processes, (Fern, Yoon and Givan), JAIR, 2006
- Learning Heuristic Functions from Relaxed Plans , (Yoon, Fern and Givan), ICAPS, 2006
- Using Learned Policies in Heuristic-Search Planning , (Yoon, Fern and Givan), IJCAI, 2007
- Goal Achievement in Partially Known, Partially Observable Domains (Chang and Amir), ICAPS, 2006
- Learning Planning Rules in Noisy Stochastic Worlds (Zettlemoyer, Pasula, and Kaelbling), AAAI, 2005
- [ARMS] Learning Action Models from Plan Examples with Incomplete Knowledge, (Yang, Wu and Jiang), ICAPS, 2005

References

- Towards Model-lite Planning: A Proposal For Learning & Planning with Incomplete Domain Models, (Yoon and Kambhampati), 2007, ICAPS-Workshop for Learning and Planning
- Markov Logic Networks (Richardson and Domingos), 2006, MLJ
- Learning Recursive Control Programs for Problem Solving (Langley and Choi), 2006, JMLR
- [HDL] Learning to do HTN Planning (Ilghami, Nau and Munoz-Avila), 2006, ICAPS
- Task Decomposition Planning with Context Sensitive Actions (Barrett), 1997