# Planning by Rewriting

**José-Luis Ambite**

University of Southern California

Information Sciences Institute and

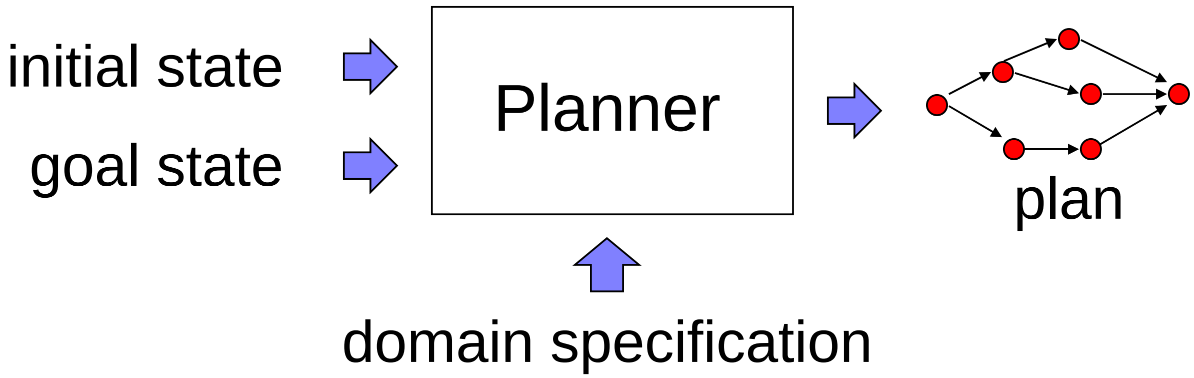
Computer Science Department

# Outline

Planning by Rewriting (PbR): A new paradigm for efficient high-quality domain-independent planning

- Motivation and Thesis Statement
- Planning by Rewriting as Local Search
- Query Planning in Mediators
- Experimental Results
- Related Work
- Future Work
- Contributions

# Domain-Independent Planning




- Many practical problems can be cast as planning
- Domain independence => Flexibility, Reusability
- But it is computationally hard [Bylander 94, Erol et al. 95]
- Moreover, plan quality is also critical

PbR addresses planning efficiency and plan quality
in a domain-independent framework

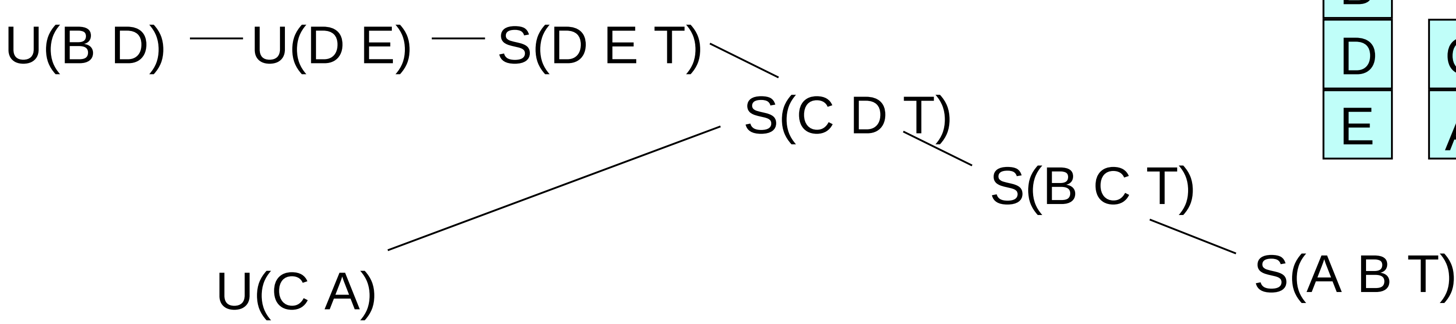
# Planning = Satisfiability + Optimization

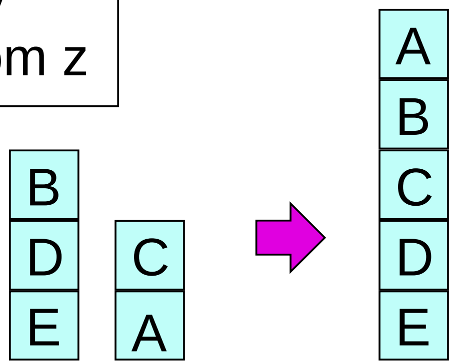
- Two sources of complexity in planning:
  - satisfiability: find *any* valid plan
  - optimization: find the *best* plan (wrt given cost metric)
- Optimization domains:
  - dominated by optimization complexity
  - finding a valid plan is easy (polynomial)
  - many practical domains:
    - query planning
    - manufacturing operations
    - transportation ...

# Transforming a suboptimal plan

U(x y): Unstack x from y
S(x y z): Stack x on y from z

**Initial Plan:**

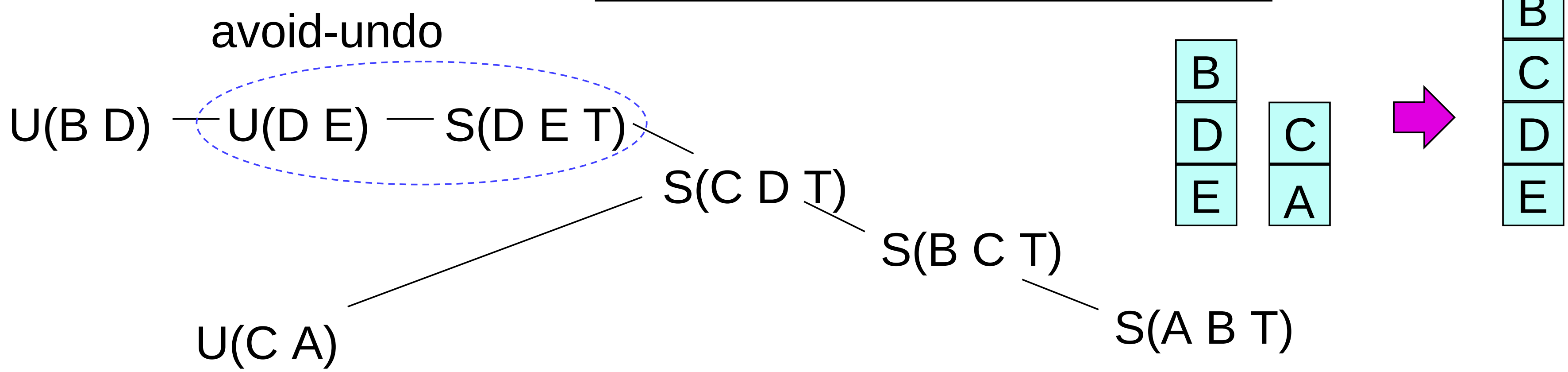


# Transforming a suboptimal plan

U(x y): Unstack x from y
S(x y z): Stack x on y from z

**Initial Plan:**



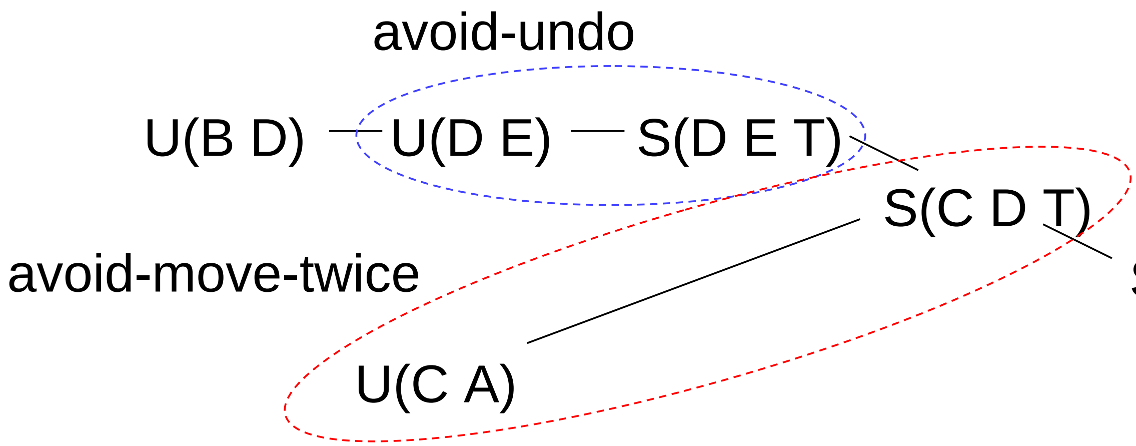

Transformations:

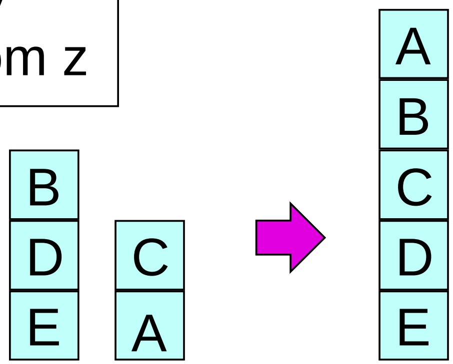
- avoid-undo: U(x y) --- S(y x T) => 0

# Transforming a suboptimal plan

U(x y): Unstack x from y
S(x y z): Stack x on y from z

**Initial Plan:**



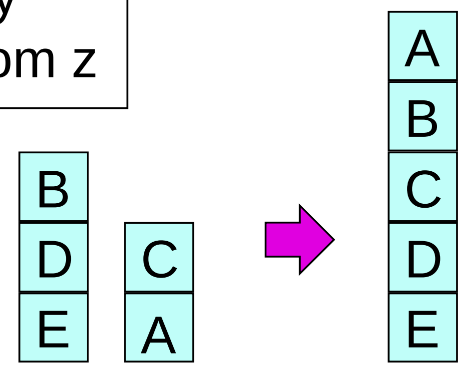




Transformations:

- avoid-undo: U(x y) --- S(y x T) => 0
- avoid-move-twice: U(x y) --- S(x z T) => S(x z y)

# Transforming a suboptimal plan

U(x y): Unstack x from y
S(x y z): Stack x on y from z




**Initial Plan:**

avoid-undo

U(B D) — U(D E) — S(D E T)

S(C D T)

avoid-move-twice

U(C A)

S(B C T)

S(A B T)

Transformations:

- avoid-undo: U(x y) --- S(y x T) => 0
- avoid-move-twice: U(x y) --- S(x z T) => S(x z y)

**Rewritten Plan:**

U(B D) — S(C DA) — S(B C T) — S(A B T)

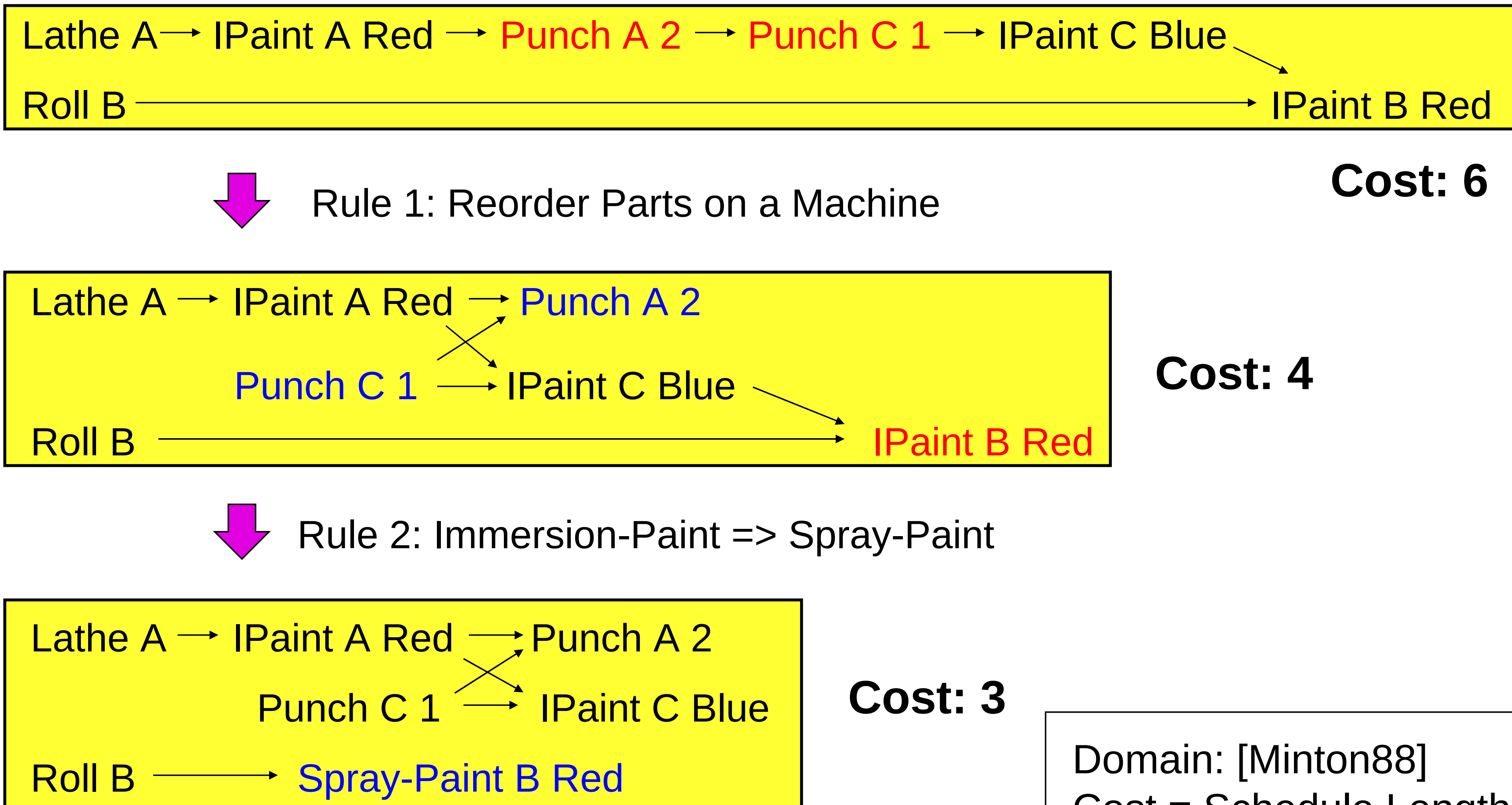
# New approach: Planning by Rewriting

- Efficiently generate an initial solution plan (possibly of low quality)
- Iteratively rewrite the current plan
  - using a set of plan rewriting rules
  - improving plan quality
  - until an acceptable solution or resource limit reached

⇓

Efficient High-Quality Planning

# Manufacturing Operations Planning




Domain: [Minton88]
Cost = Schedule Length

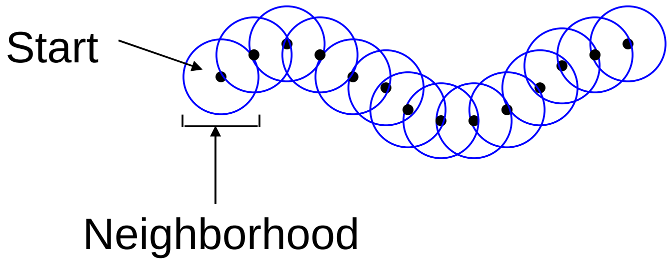
# Thesis Statement

Declarative plan rewriting
combined with local search provide

Efficient High-Quality
Domain-Independent Planning

# Planning by Rewriting as Local Search

- PbR: efficient high-quality planning using local search
- Main issues:
  - *Selection of initial feasible point*: Initial plan generation.
  - *Generation of a local neighborhood*: Set of plans obtained from application of the plan rewriting rules.
  - *Cost function to minimize*: Measure of plan quality.
  - *Selection of next point*: Next plan to consider. Determines how the global space is explored.

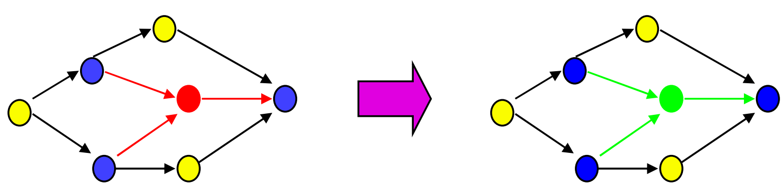


# Generation of an Initial Plan

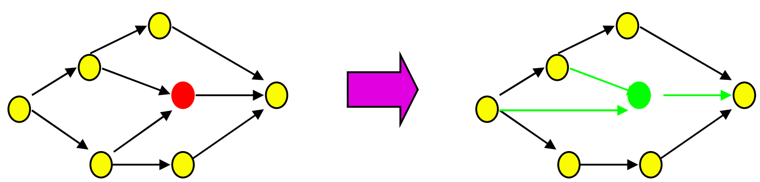
- Biased generative planners
  - Domain-independent: HSP (heuristic search)
  - Domain-specific search control rules:
    - Directly specified: UCPOP, TLPlan (temporal logic)
    - Learning, abstraction: Prodigy, IPP-GAM
  - Example: process planning, depth-first search and search control automatically generated by an abstraction hierarchy
- Programmatically
  - Approximation algorithms. Examples:
    - query planning: any parse of the query (or a greedy one)
    - blocksworld: *"put all blocks in the table, then build towers"* (linear)
  - Provided high-level plan construction language

# Generation of a Local Neighborhood

- Declarative plan rewriting rules: express concisely complex transformations
- More natural than search control: complete plan and cost
- Intention: Move towards higher quality solutions
- Result of a rewriting rule is always a solution plan
- Two types of rewriting rules:
  - Fully-specified: typical of graph rewriting



  - Partially-specified: exploit semantics of planning



# Partially-Specified Rewriting Rules

- Embedding of rule consequent is not explicit in rule antecendent
- Uses semantics of partial order planning to compute the embedding

```
(define-rule :name avoid-move-twice
  :if (:operators ((?n1 (unstack ?b1 ?b2))
                   (?n2 (stack ?b1 ?b3 Table)))
       :links (?n1 (on ?b1 Table) ?n2)
       :constraints ((possibly-adjacent ?n1 ?n2)
                     (:neq ?b2 ?b3)))
  :replace (:operators (?n1 ?n2))
  :with (:operators (?n3 (stack ?b1 ?b3 ?b2))))
```

subplans

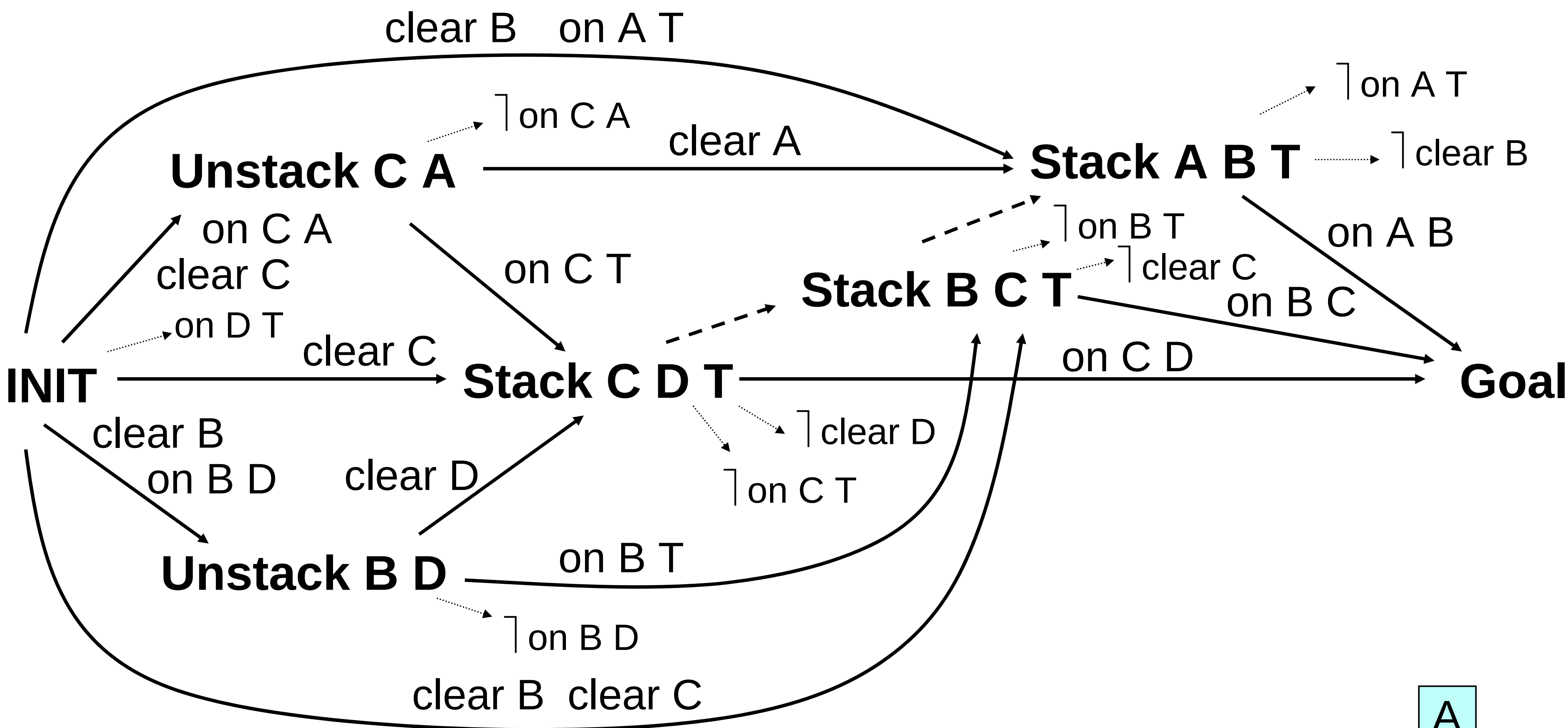
# Fully-Specified Rewriting Rules

- Embedding of rule consequent fully specified
- All anchors present in antecedent

```
(define-rule :name avoid-move-twice-full-spec
  :if (:operators ((?n3 (unstack ?b1 ?b2))
                   (?n8 (stack ?b1 ?b3 Table)))
       :links ((?n1 (clear ?b1) ?n3) (?n2 (on ?b1 ?b2) ?n3)
               (?n3 (clear ?b2) ?n4) (?n3 (on ?b1 Table) ?n5)
               (?n6 (on ?b1 ?b3) ?n9) (?n7 (clear ?b1) ?n9)
               (?n8 (clear ?b3) ?n9) (?n9 (on ?b1 ?b3) ?n10))
       :constraints ((possibly-adjacent ?n3 ?n8) (:neq ?b2 ?b3)))
  :replace (:operators (?n1 ?n2))
  :with (:operators ((?n11 (stack ?b1 ?b3 ?b2)))
         :links ((?n1 (clear ?b1) ?n11)
                 (?n8 (clear ?b3) ?n11)
                 (?n2 (on ?b1 ?b2) ?n11)
                 (?n11 (on ?b1 ?b3) ?n10
```

subplans

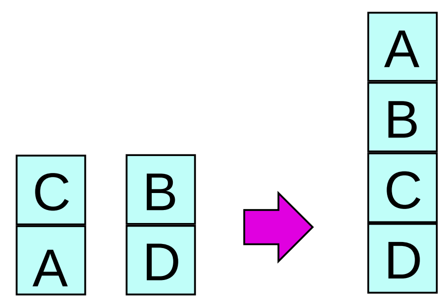
# Plan Rewriting



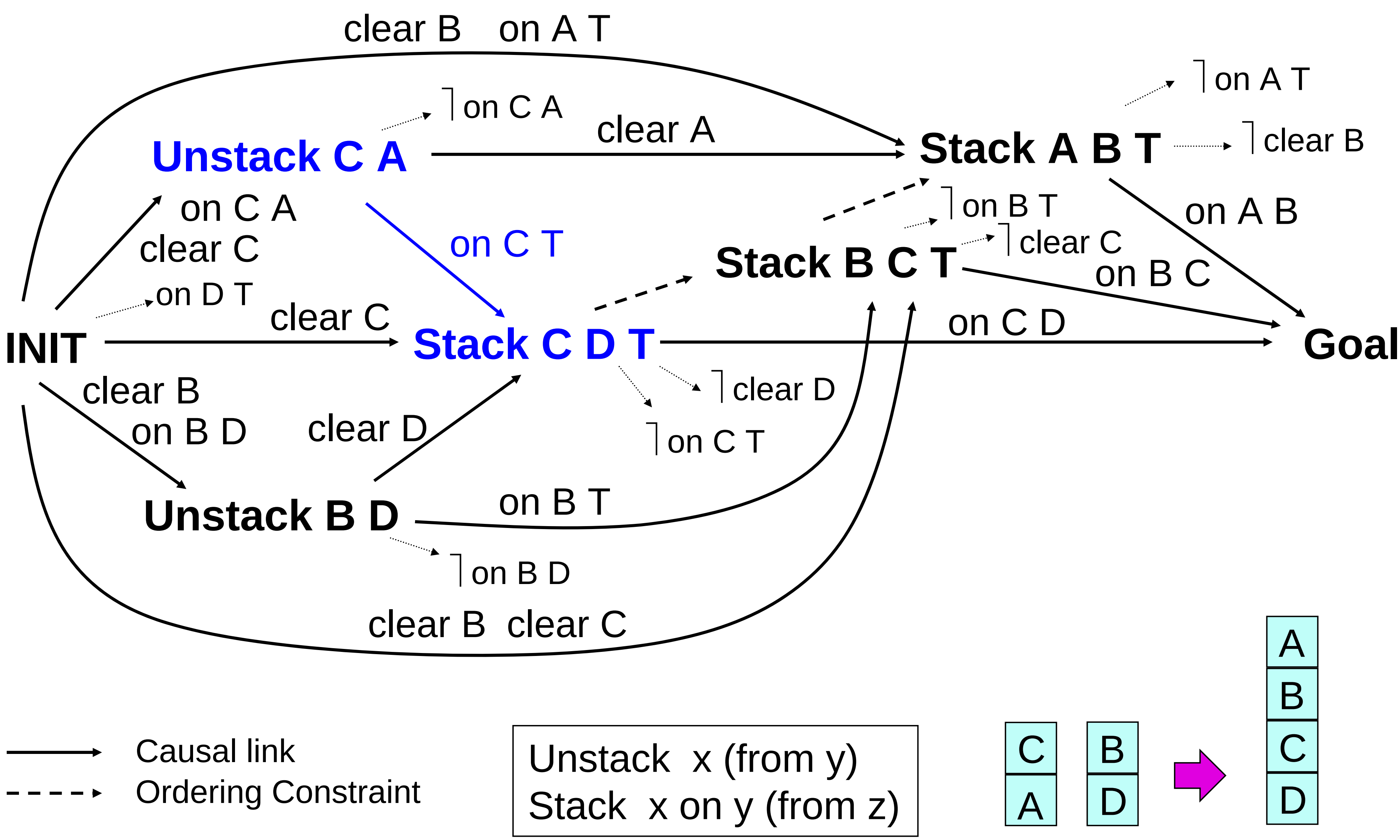

Causal link

Ordering Constraint

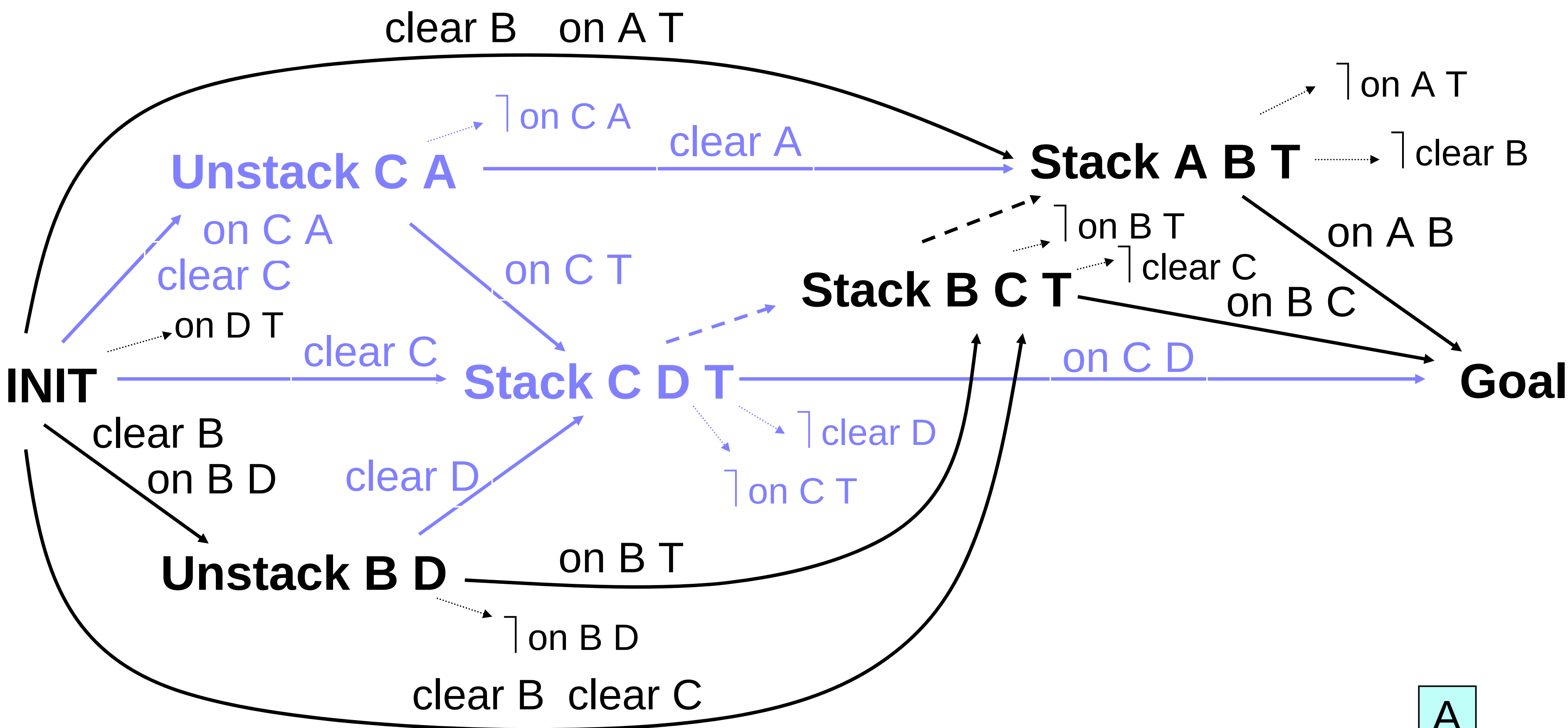
Unstack x (from y)
Stack x on y (from z)



# Plan Rewriting



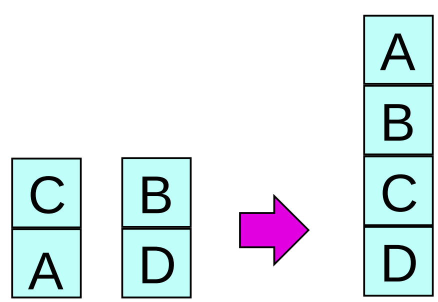
# Plan Rewriting



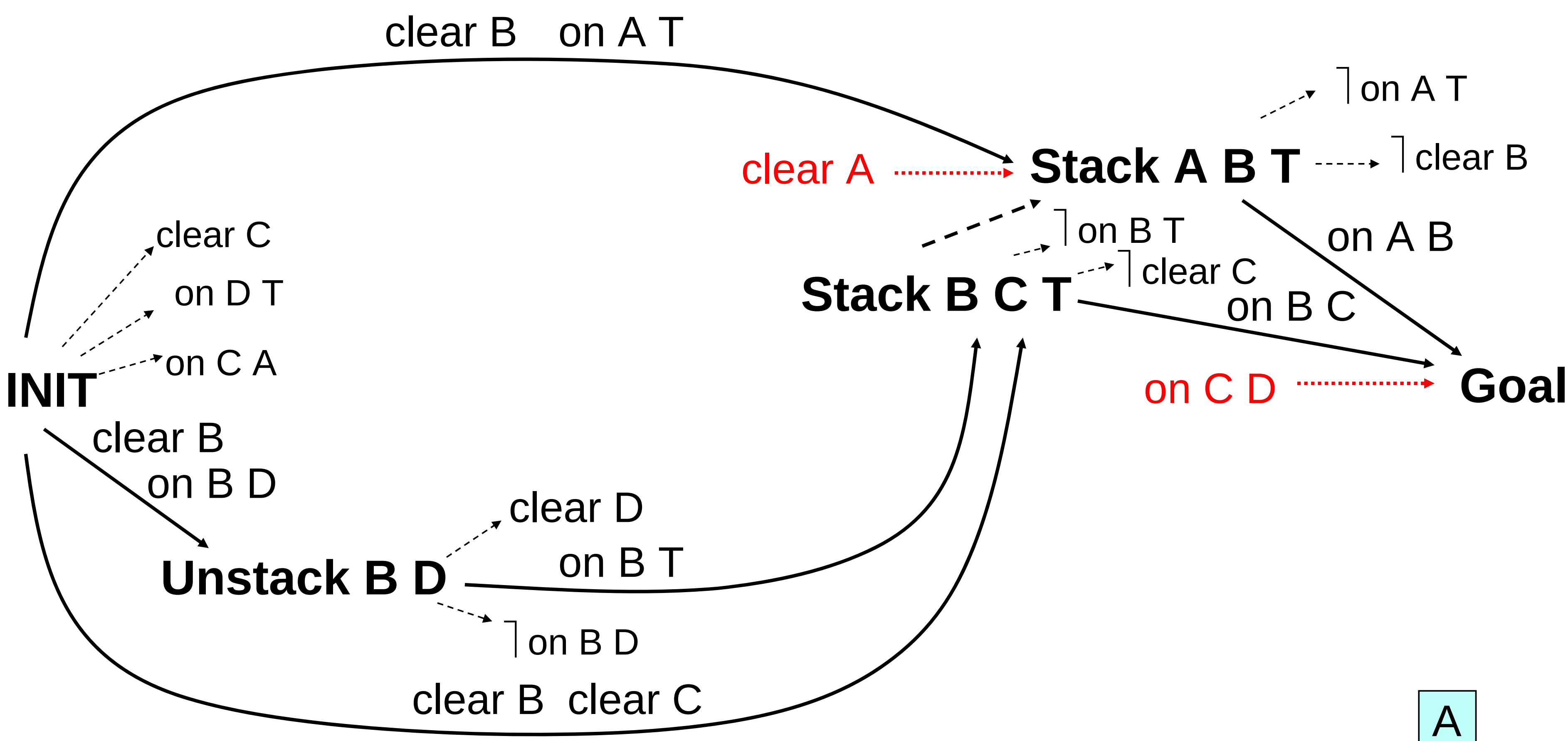

Causal link

Ordering Constraint

Unstack x (from y)
Stack x on y (from z)

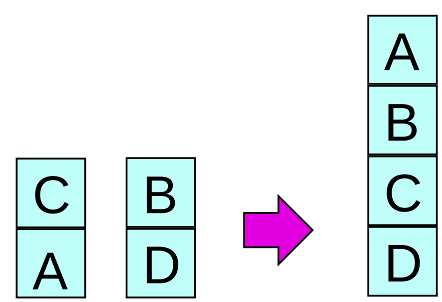


# Plan Rewriting

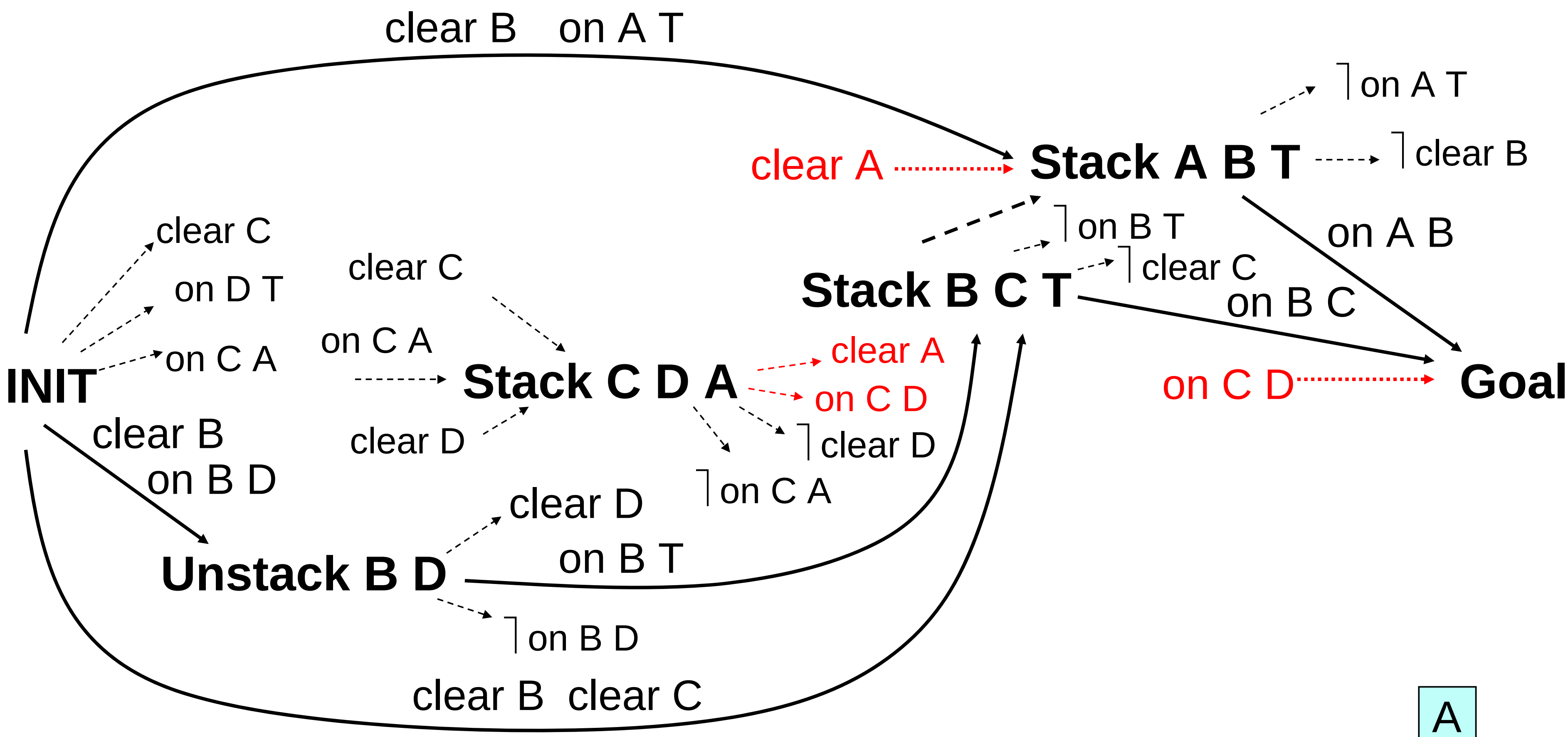



Causal link
Ordering Constraint

Unstack x (from y)
Stack x on y (from z)



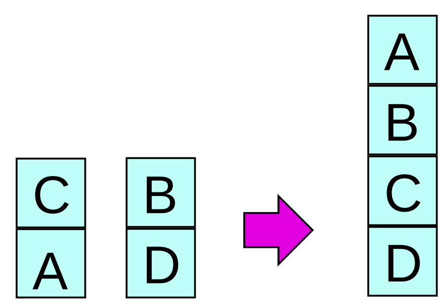
# Plan Rewriting



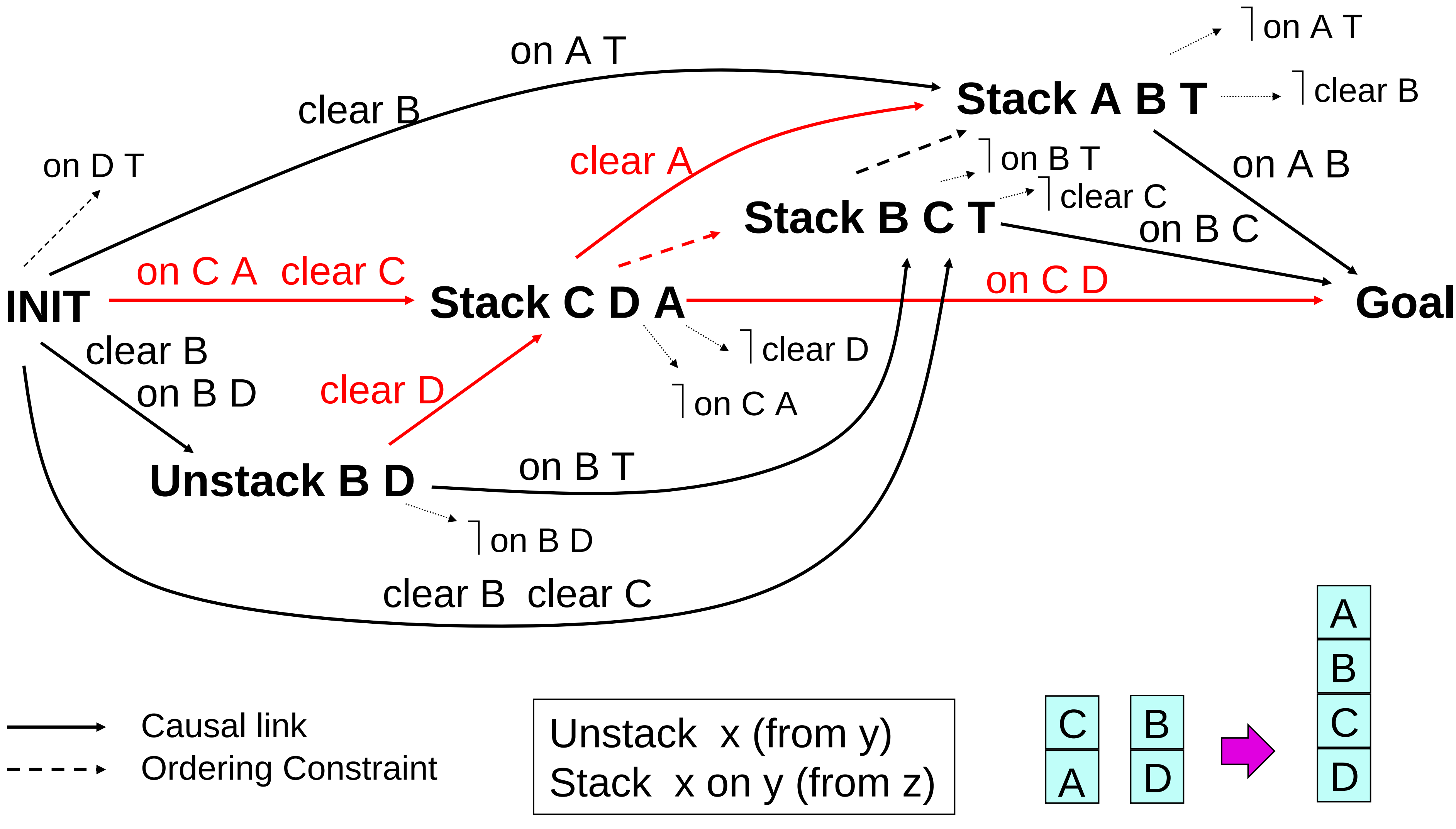

Causal link

Ordering Constraint

Unstack x (from y)
Stack x on y (from z)



# Plan Rewriting




Unstack x (from y)
Stack x on y (from z)

# Selection of Next Plan

- Determines search in the solution space. Affects:
  - quality of the solution
  - rate of convergence
- Explored gradient-descent techniques:
  - first improvement: partially explores neighborhood, but smaller improvement
  - steepest-descent: explores complete neighborhood, but greatest improvement
  - to escape local minima:
    - restart from different/random initial plans
    - random walk (in plateaus)

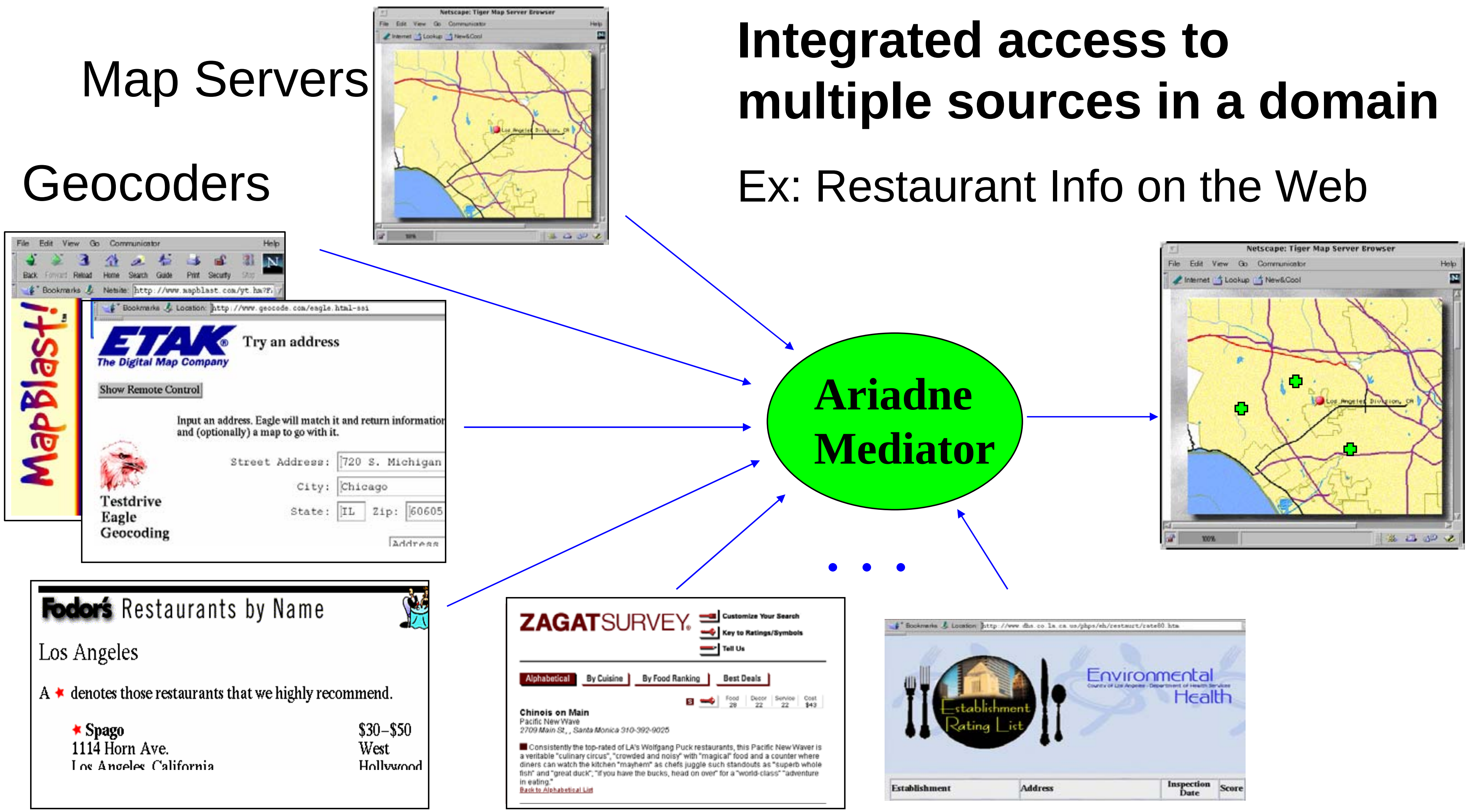
# Application of PbR: Query Planning in Mediator Systems

**Integrated access to multiple sources in a domain**

Ex: Restaurant Info on the Web

Map Servers

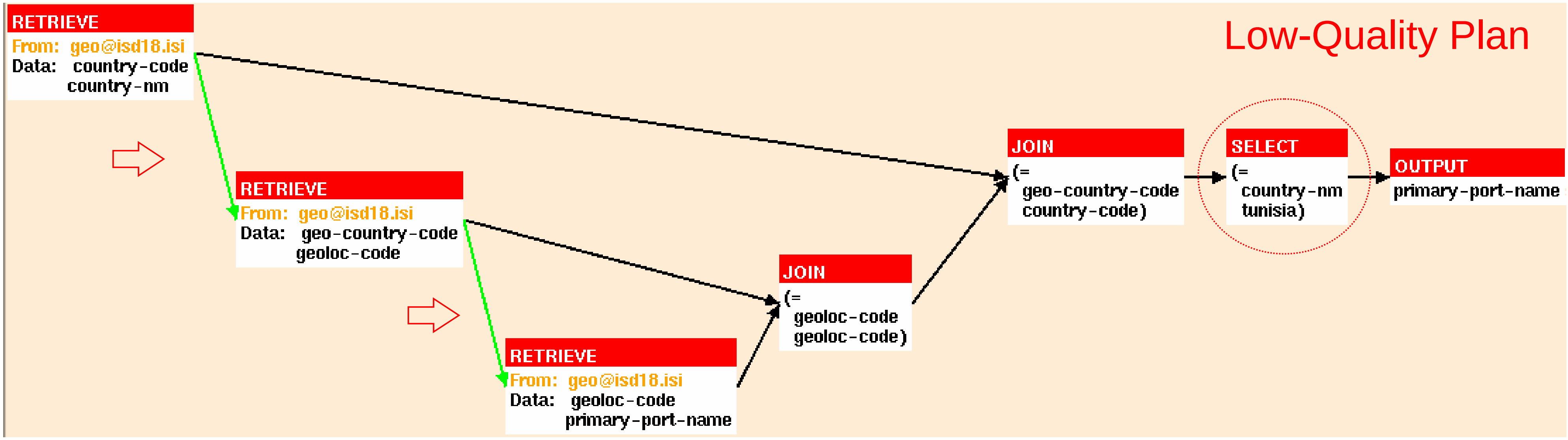
Geocoders

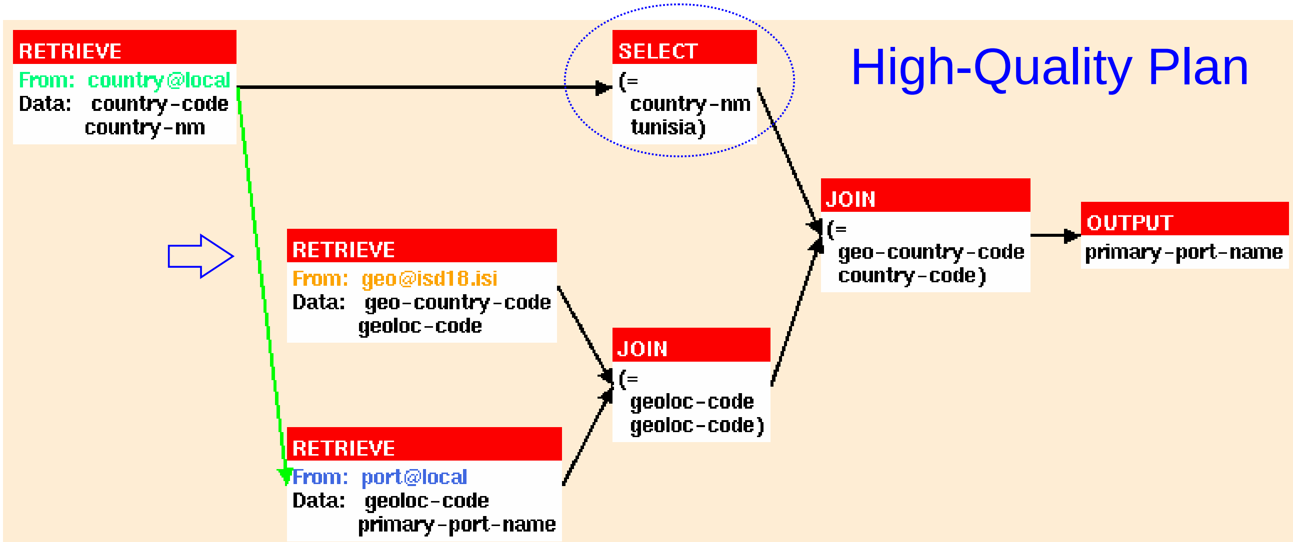



Fodor's Zagat Health Ratings

José-Luis Ambite

# Query Plans and Plan Quality



# Planning by Rewriting for Query Planning in Mediators

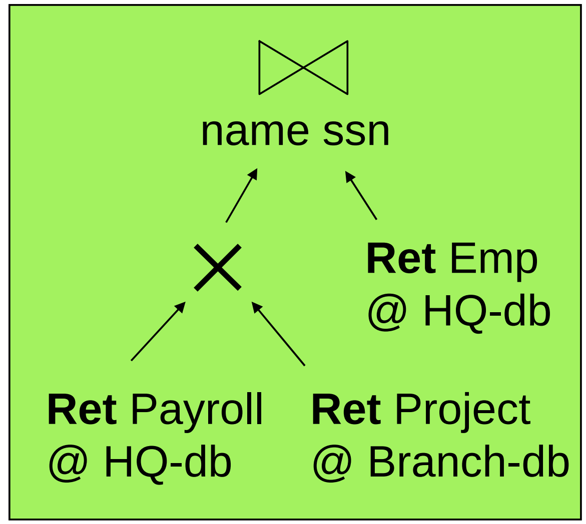
- *Initial plan generation*: random parse of the query
- *Plan rewriting rules*: based on properties of:
  - relational algebra,
  - distributed environment,
  - integration axioms
- *Plan quality:* query execution time (size estimation)
- *Search Strategies:* gradient descent+restart, simulated annealing, variable-depth rewriting, ...

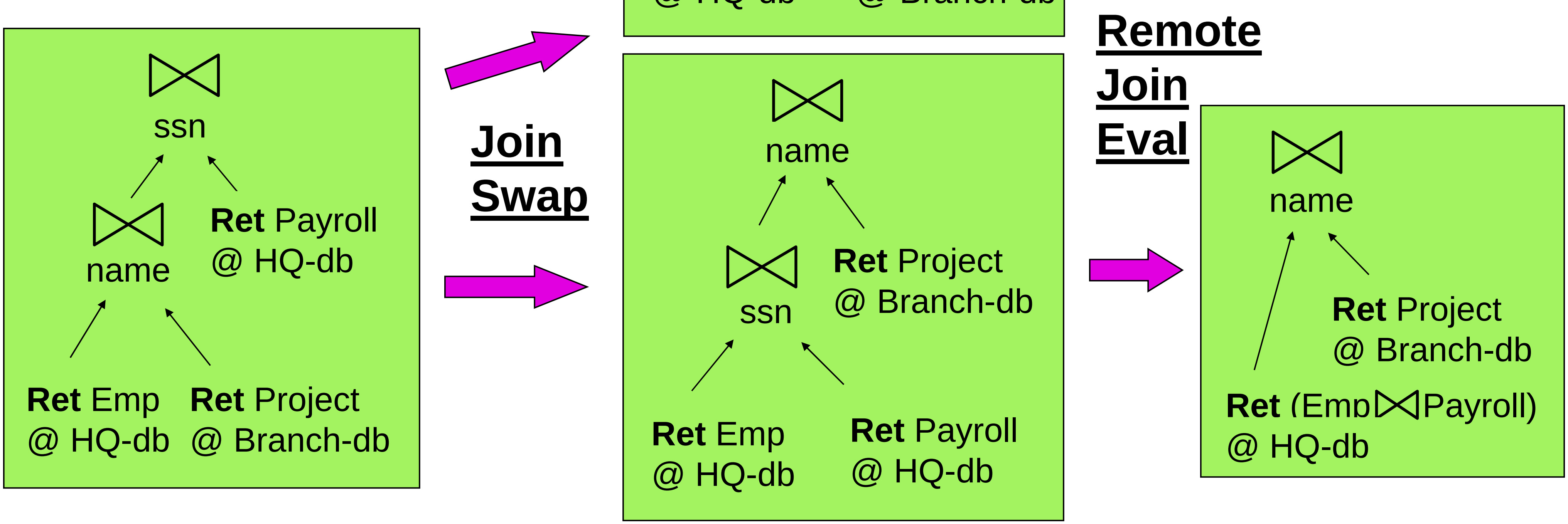
# Query Planning in PbR

a(name sal proj) :- Emp(name ssn) ^
Payroll(ssn sal) ^
Projects(name proj)

| HQ-db |
| --- |
| Emp(name ssn) |
| Payroll(ssn sal) |

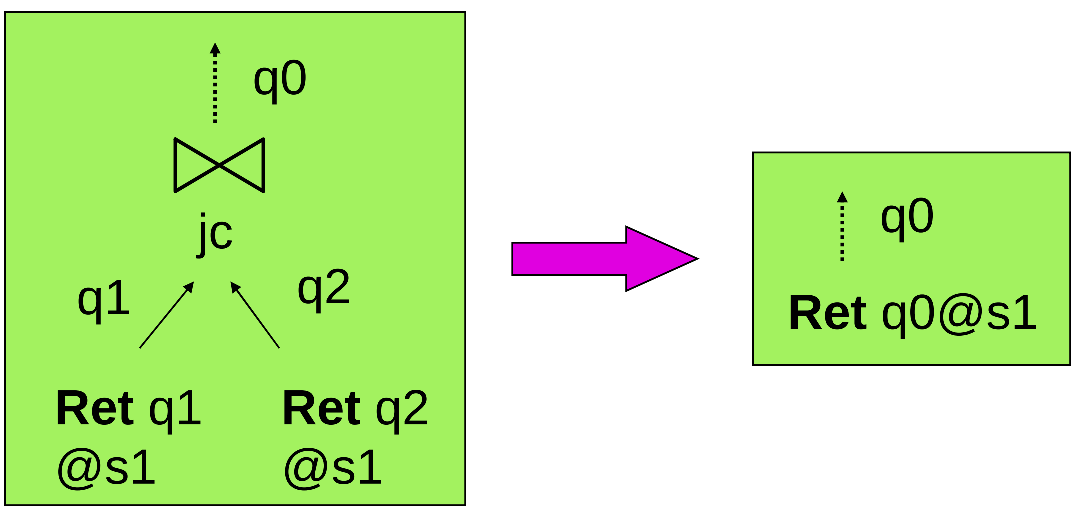
| Branch-db |
| --- |
| Project(name proj) |



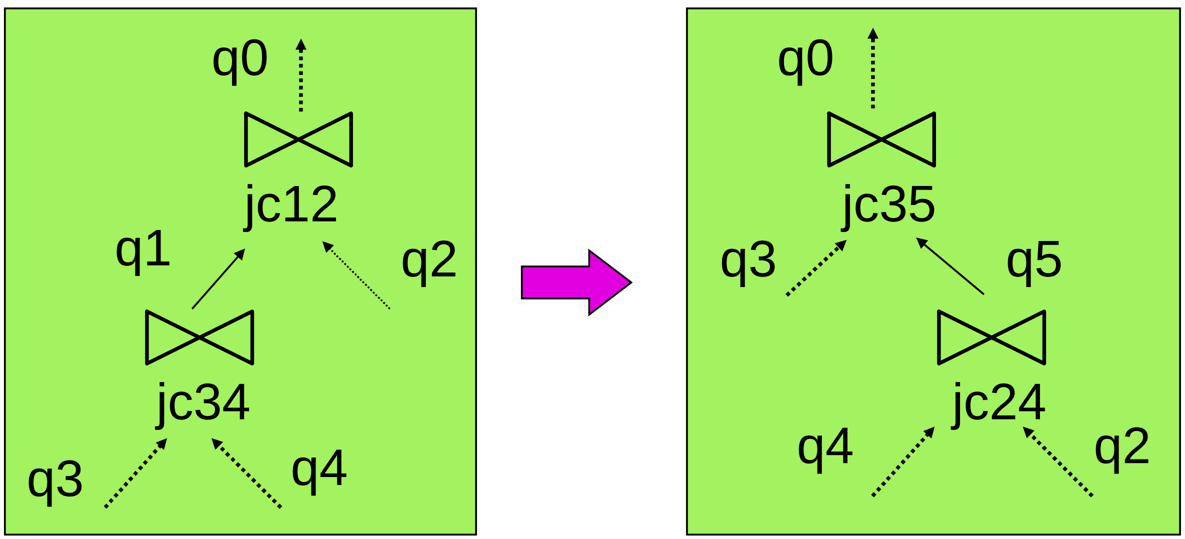
# Rewriting Rules: Distributed Environment
# **remote-join-eval**

```
(define-rule :name remote-join-eval
  :if (:operators ((?n1 (retrieve ?source ?query1))
                   (?n2 (retrieve ?source ?query2)
                   (?n3 (join ?join-conds ?query0 ?query1 ?query2)))
       :constraints (capability ?source join))
     :replace (:operators (?n1 ?n2 ?n3))
     :with (:operators ((?n4  (retrieve ?source ?query0)))))
```



# Rewriting Rules: Relational Algebra
# **join-associativity**

```
(define-rule :name join-associativity
  :if (:operators ((?n1 (join ?jc34 ?q1 ?q3 ?q4)
              (?n2 (join ?jc12 ?q0 ?q1 ?q2)))
     :constraints (join-swappable ?jc34 ?q1 ?q3 ?q4 ?jc12 ?q0 ?q2 ;; in
                          ?jc24 ?jc35 ?q5))          ;; out
  :replace (:operators (?n1 ?n2))
  :with (:operators ((?n3 (join ?jc24 ?q5 ?q4 ?q2))
               (?n4 (join ?jc35 ?q0 ?q3 ?q5)))
```



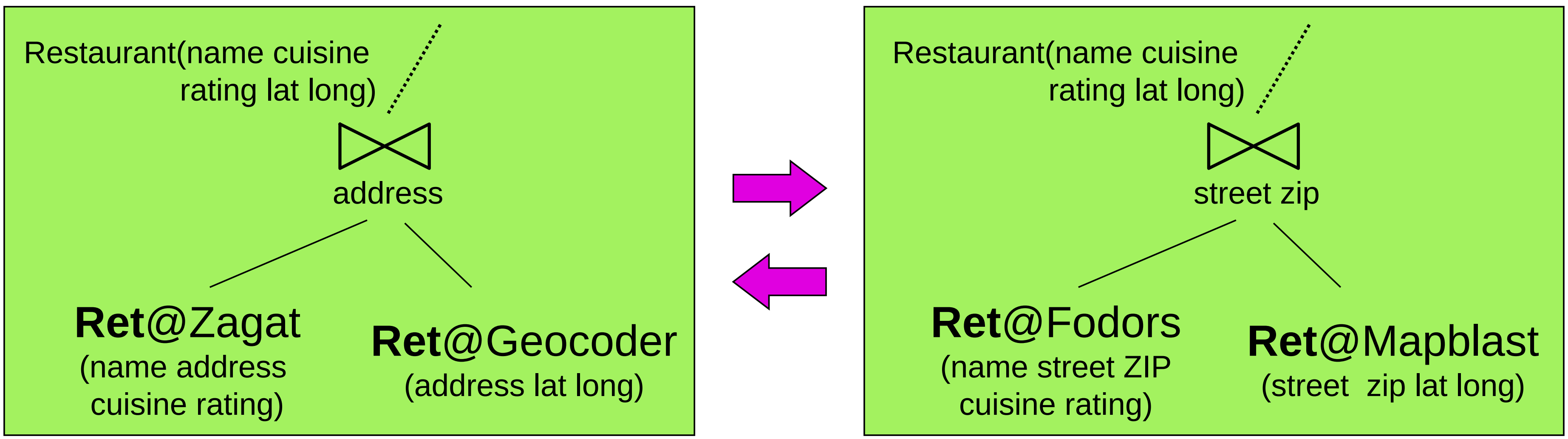
# Rewriting Rules: Integration Axioms

- Rules computed from integration axioms relevant to query:

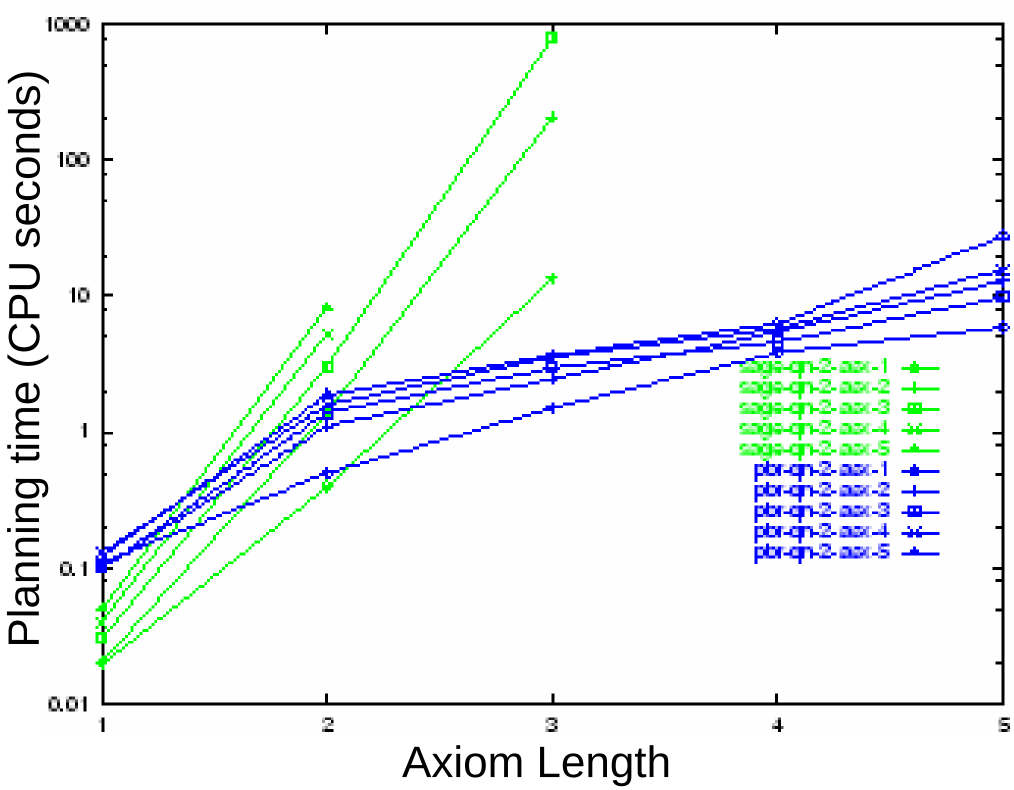
Restaurant(name cuisine rating lat long) =

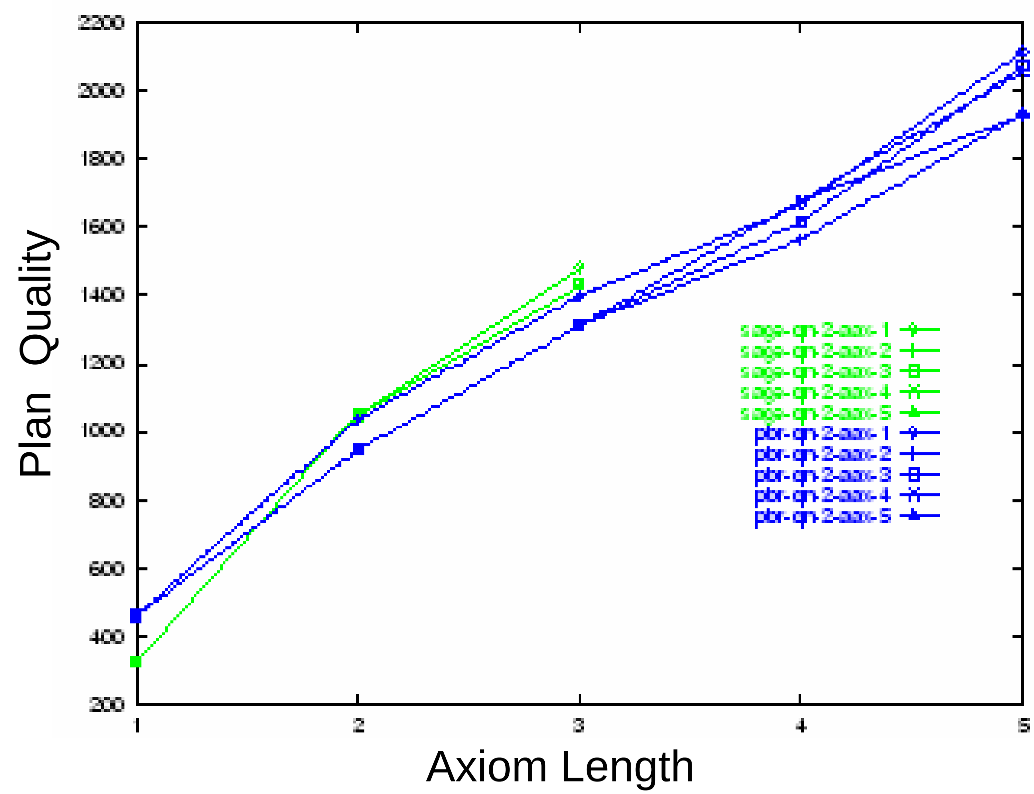
a) Zagat(name address cuisine rating) ^ Geocoder(address lat long)

b) Fodors(name street zip cuisine rating) ^ Mapblast(street zip lat long)



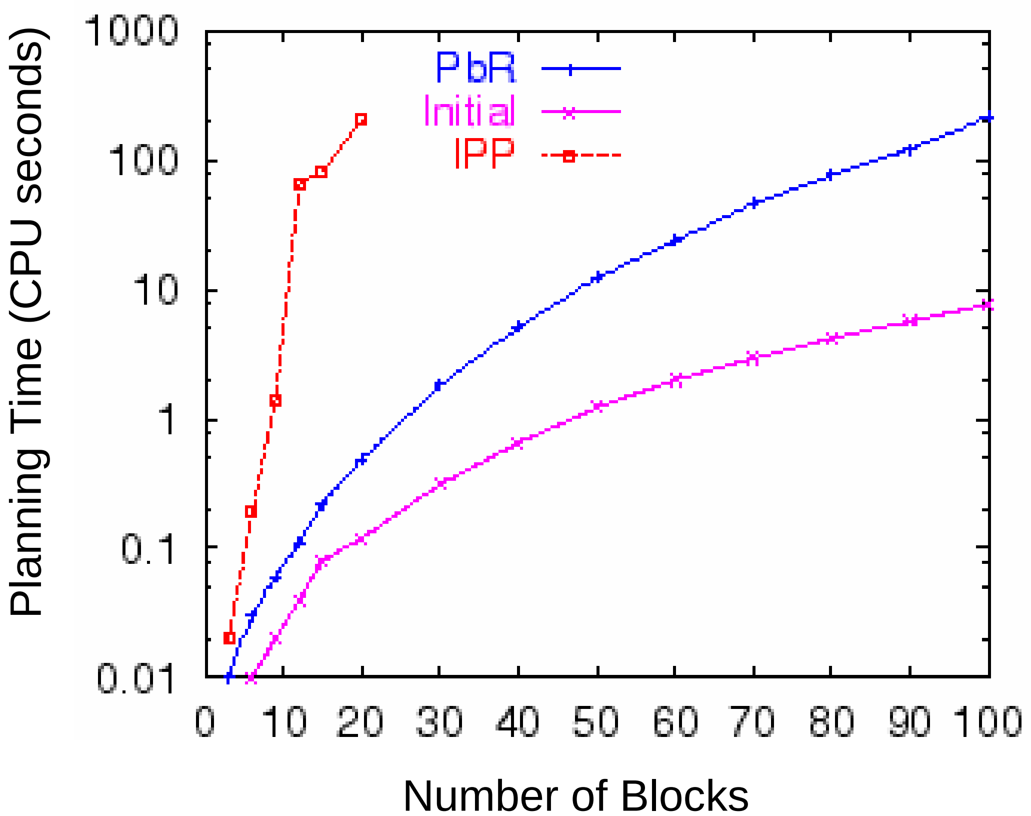
# Scaling Axiom Length and Number of Alternative Axioms

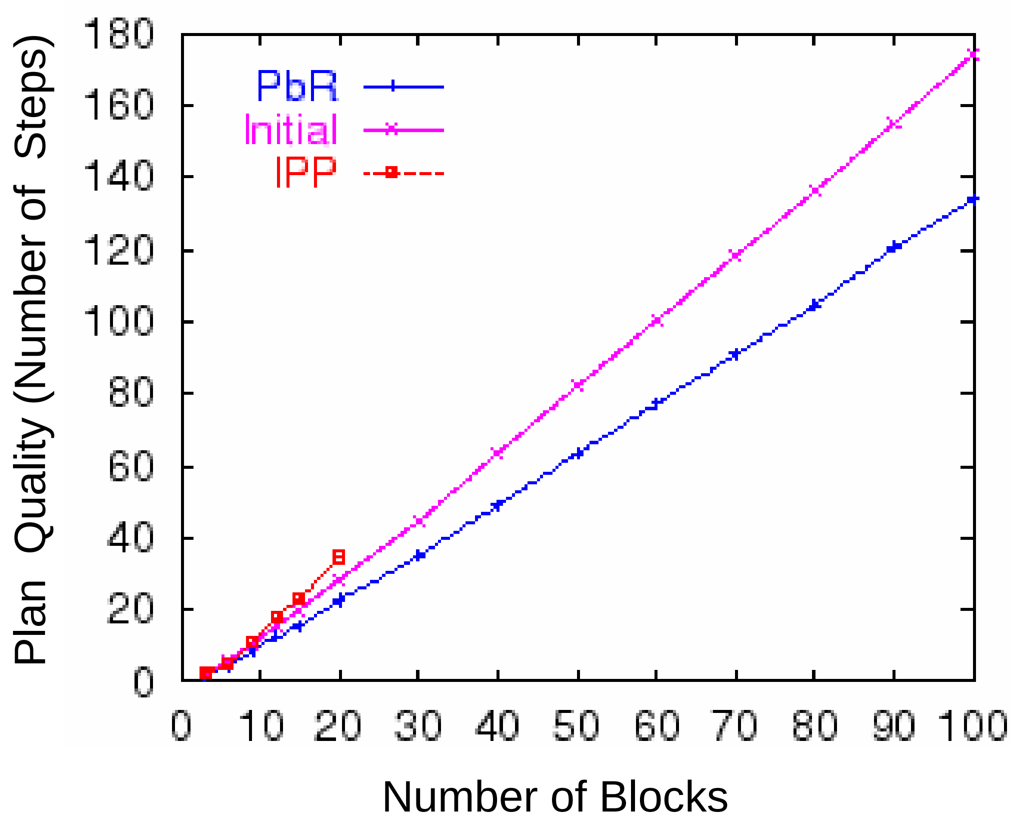



- Query planning in mediators
  - PbR is scalable
  - PbR produces high-quality plans

# PbR vs State-of-the-Art (IPP)




- Blocksworld
  - PbR is more scalable than IPP
  - PbR produces higher-quality plans than IPP

# Limitations

- No guarantee of optimality
- Initial plan generator:
  - User specified
  - Empirically, efficient (suboptimal) planners
- Rewriting rules:
  - User specified
  - More natural than search control
  - Learning is possible

# Related Work (General)

- Planning Efficiency
  - Learning Search Control [Minton 88][Knoblock 94][Etzioni94]
  - Planning as satisfiability + stochastic search [Selman 96]
- Plan Quality  [Perez 96]
- Local Search [Papadimitriou & Steiglitz 82] [Aarts& Lenstra 97]
  - Constraint Satisfaction, scheduling [Minton 92] [Zweben+94]
  - Heuristic Search [Ratner & Pohl 86]
- Graph Rewriting [Schurr 96]
- Plan Rewriting:
  - Plan Merging  [Foulser, Li & Yang 92]
  - Case-based Planning [Hanks & Weld 95] [Veloso 94]

# Related Work (Query Planning)

- Traditional Query optimization
  - Distributed Query Optimization: [Chu&Hurley 82]
  - Extensible Query Optimizers: Starburst [Pirahesh et al 92] Exodus[Graefe & DeWitt 87] Volcano [Graefe 93]
  - Efficient Search: [Swami 89] [Ioannidis & Kang 90]
- Query Planning in Mediators
  - IM [Levy et al 96]  TSIMMIS [Hammer et al 95]
  - HERMES [Adali et al 96]
  - Garlic [Hass et al 97]
- Query Planning in AI planning: Occam [Kwok&Weld96] Sage [Knoblock95]

# Contributions

- Planning by Rewriting: Efficient high-quality domain-independent planning
  - Plan rewriting rules (fully-specified and partially-specified): naturally concisely express complex plan transformations
  - Plan rewriting algorithm
  - Scalable using local search
  - Anytime behavior
- PbR-based query planner for mediators
  - Declarative: Flexible, Extensible, Reusable
  - Combines cost-based query optimization and source selection

# Future Work

- Learning
  - Rewriting Rule Generation: static analysis, example based
  - MultiTAC-like system [Minton 93]: automatic configuration
- Search Strategies: many ideas from local search
  - Ex: variable depth rewriting: rule programs
- Rewriting through incomplete plans: subsumes generative planning (linear, partial-order, and HTN)
- Query Planning:
  - Interplay of rewriting and execution: run-time info
  - Source capabilites (binding patterns
  - New transformations: extend language, physical operators

# END