Meta-Reasoning for Self-Adaptation

Ashok K. Goel

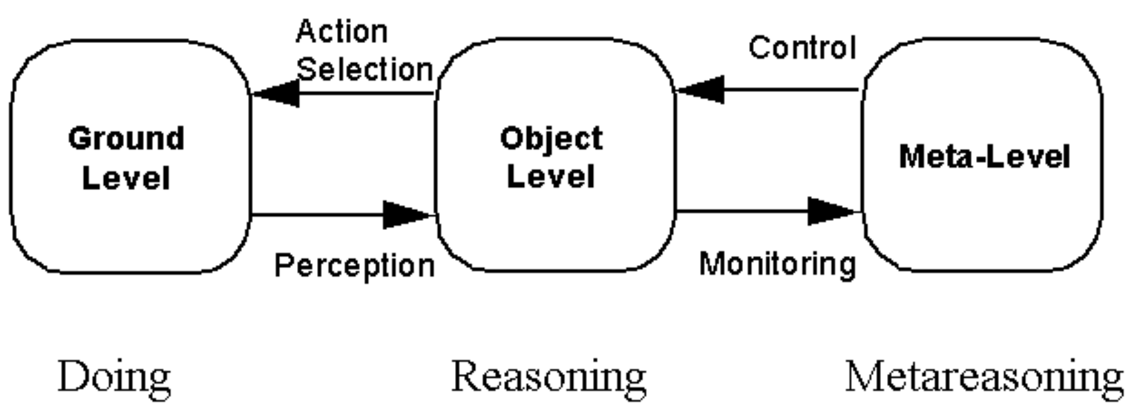
Design & Intelligence Laboratory
Georgia Institute of Technology

AAAI-2010 Workshop on Meta-Cognition, Atlanta, July 2010

Outline:

1. *Introduction: why, what, when and how of meta-reasoning.*
2. Proactive, goal-directed reconfiguration of reasoning processes.
3. Combining model-based meta-reasoning and reinforcement learning.
4. Retrospective, failure-driven repair of domain knowledge.
5. Conclusions: model-based meta-reasoning for self-adaptation.

Metareasoning



Basic metareasoning architecture
Adapted from Cox & Raja, 2007

Why Meta-Reasoning?

Control of reasoning (e.g., Hayes-Roth, Raja & Lesser, Zilberstein)
Bounding of reasoning (e.g., Russell, Horvitz)

Self-Explanation (e.g., Brueker & Wilenga, Chandrasekaran,
Cox & Ram, Leake, Murdock & Goel)

Retrospective, failure-driven revision of beliefs (e.g., Doyle)
Retrospective, failure-driven revision of domain knowledge
(e.g., Davis, Leake)
Retrospective, failure-driven revision of reasoning processes
(e.g., Anderson, Josyula, Oates & Perlis,
Cox & Ram, Freed & Birnbaum, Stroulia & Goel)

Proactive, goal-directed revision of reasoning processes
Localization of situated learning

When Meta-Reasoning (for Self-Adaptation)?

Retrospective, failure-driven self-adaptation.

Proactive, goal-directed self-adaptation.

What Meta-Reasoning (for Self-Adaptation)?

About the situated element.
(e.g., Anderson, Josyula, Oates & Perlis, Stroulia & Goel).

About the deliberative reasoner.
About the reasoner's beliefs.
About the reasoner's domain knowledge.
About the reasoner's reasoning processes.

How Meta-Reasoning (for Self-Adaptation)?

Libraries of failures, faults, fixes (e.g., Sussman)

Explanation patterns (e.g., Cox & Ram, Leake)

Traces of processing (e.g., Haigh & Veloso)

Models of the agent (e.g., Freed & Birnbaum, Stroulia & Goel)

> *A model of the agent's design enables self-adaptation.*
> *Model-based meta-reasoning.*

Outline:

1. Introduction: model-based meta-reasoning for self-adaptation.

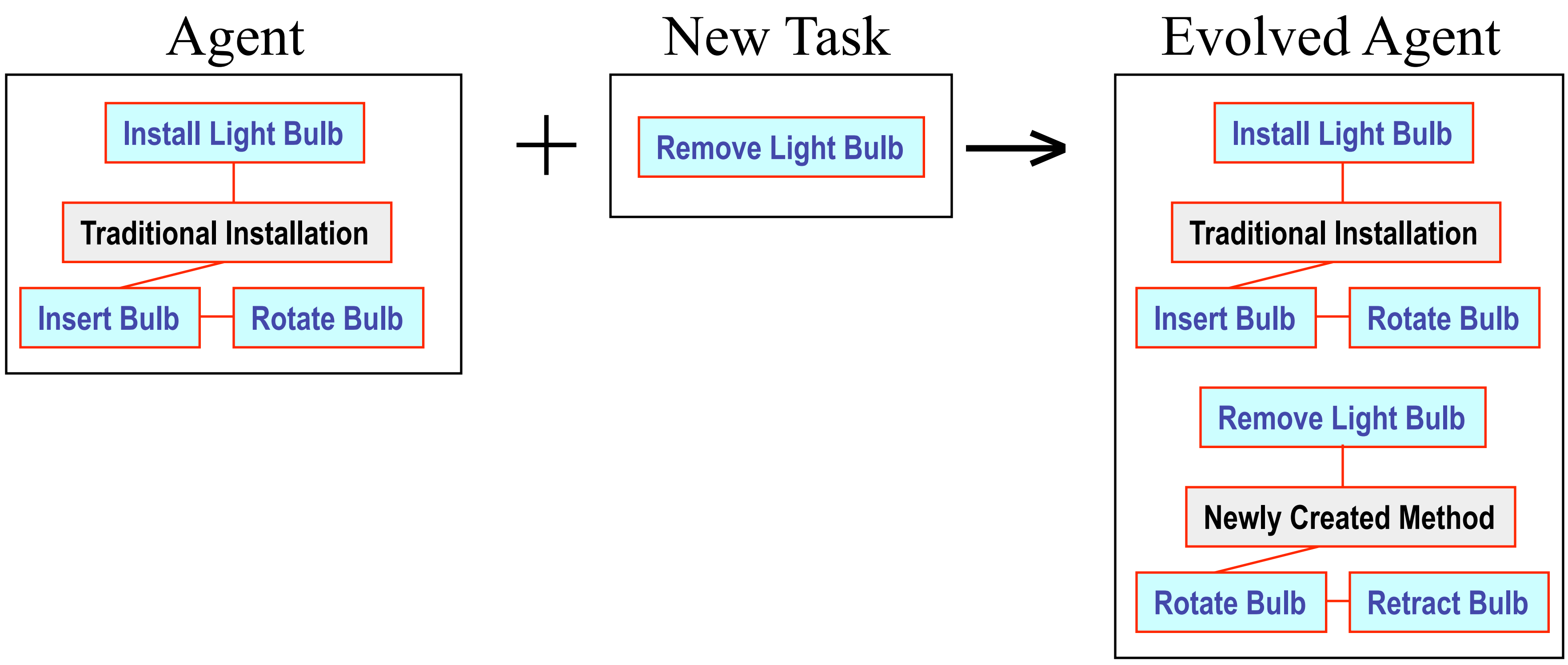
2. *Proactive, goal-directed reconfiguration of reasoning processes.*
   Joint work with J. William Murdock

3. Localizing reinforcement learning.

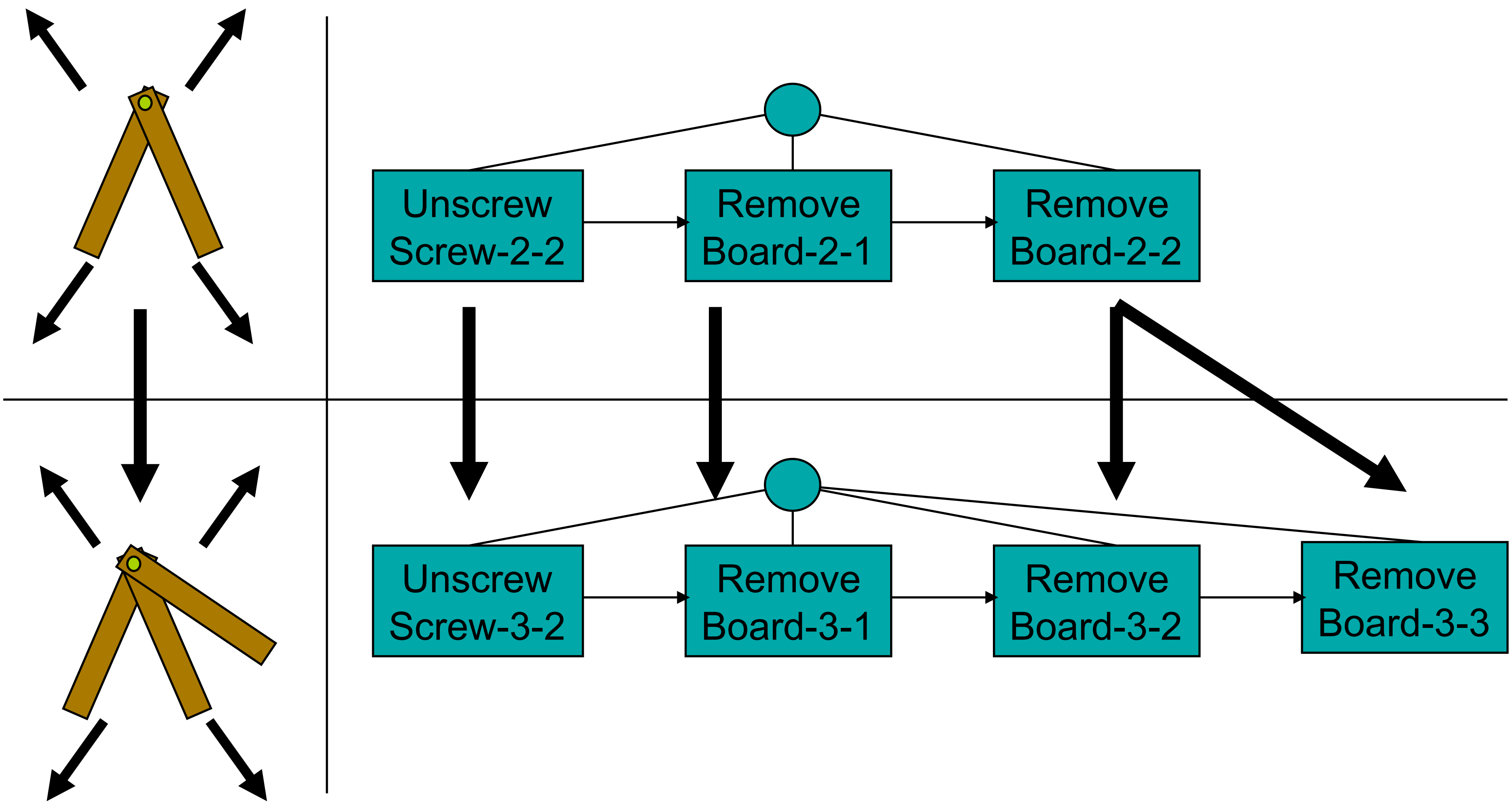
4. Retrospective, failure-driven repair of domain knowledge.

5. Conclusions: model-based meta-reasoning for self-adaptation.

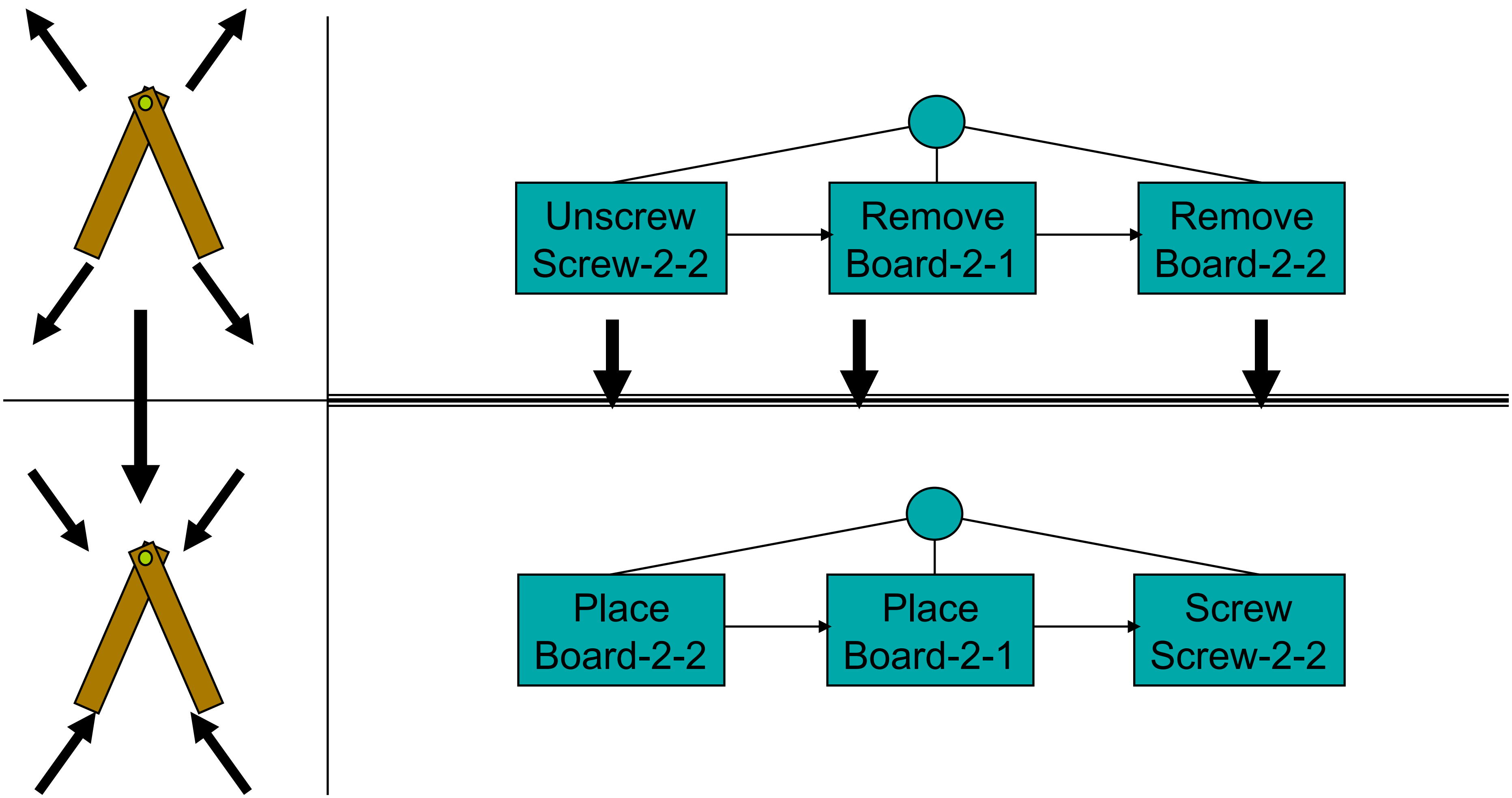
Proactive Self-Adaptation: Simple Example



Why Not Simple Analogy?
Example: Disassembly Planning



But what if causal ordering of actions is different?: Example: Assembly Problem



Different Kinds of Similarity

- When different problems have very different solutions (i.e. different causal ordering of actions), then complexity of analogical transfer of solutions is no better than of generative planning.
- However, different problems may be solved using very similar reasoning mechanisms (causal ordering of reasoning tasks, methods, etc.).
- Can we analogically transfer reasoning mechanisms?
- *Meta-Cases*, *Meta-Case-Based Reasoning*, *Meta-Analogies*

(Murdock & Goel 1996, 2001, 2008)

The REM (Reflective Evolutionary Mind) Shell

- Task Method Knowledge models may provide an agent with knowledge of its own design.

  Origin in task models of knowledge systems

  (e.g., Generic Tasks, Problem-Solving Methods, CommonKADS)
- TMKL for expressing TMK models of agent designs.
- REM can execute the reasoning processes of agents encoded in TMKL. It can also adapt them for some classes of problems.

(Murdock & Goel 2001, 2003, 2008)

Tasks in TMKL

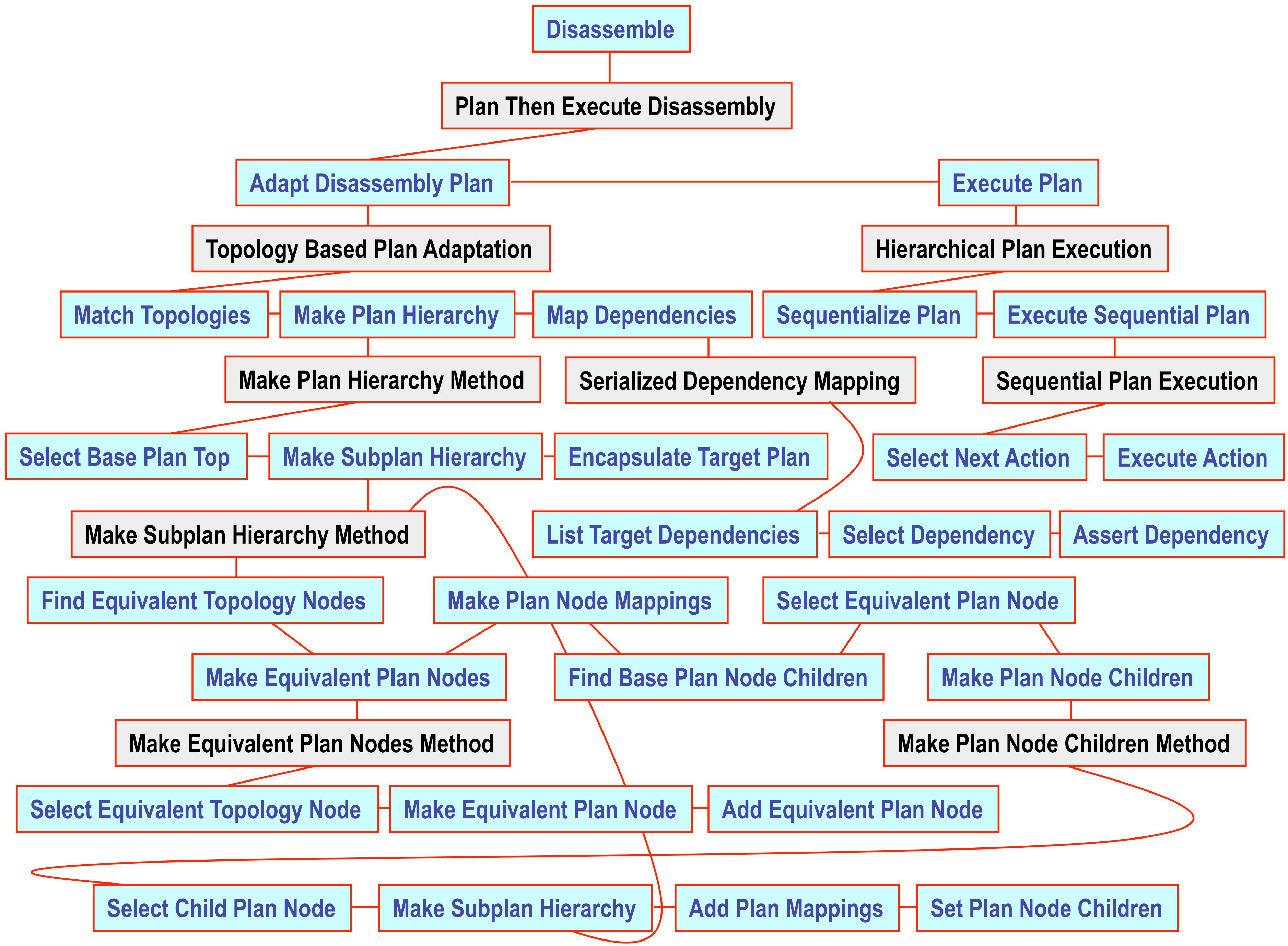
- All tasks can have input & output parameter lists and given & makes conditions.
- A non-primitive task must have one or more methods which accomplishes it.
- A primitive task must include one or more of the following: source code, a logical assertion, a specified output value.
- Unimplemented tasks have neither of these.

Methods in TMKL

- Methods have provided and result conditions.
- In addition, a method specifies a start transition for its processing control.
- Each transition specifies requirements for using it and a new state that it goes to.
- Each state has a task and a set of outgoing transitions.

Physical Device Disassembly

- ADDAM: Legacy software agent for hierarchical case-based disassembly planning and (simulated) execution
- Interactive: Agent connects to a user specifying goals and to a complex physical environment
- Dynamic: New designs and demands
- Knowledge Intensive: Designs, plans, etc.


Disassemble
Plan Then Execute Disassembly
Adapt Disassembly Plan
Execute Plan
Topology Based Plan Adaptation
Hierarchical Plan Execution
Match Topologies
Make Plan Hierarchy
Map Dependencies
Sequentialize Plan
Execute Sequential Plan
Make Plan Hierarchy Method
Serialized Dependency Mapping
Sequential Plan Execution
Select Base Plan Top
Make Subplan Hierarchy
Encapsulate Target Plan
Select Next Action
Execute Action
Make Subplan Hierarchy Method
List Target Dependencies
Select Dependency
Assert Dependency
Find Equivalent Topology Nodes
Make Plan Node Mappings
Select Equivalent Plan Node
Make Equivalent Plan Nodes
Find Base Plan Node Children
Make Plan Node Children
Make Equivalent Plan Nodes Method
Make Plan Node Children Method
Select Equivalent Topology Node
Make Equivalent Plan Node
Add Equivalent Plan Node
Select Child Plan Node
Make Subplan Hierarchy
Add Plan Mappings
Set Plan Node Children

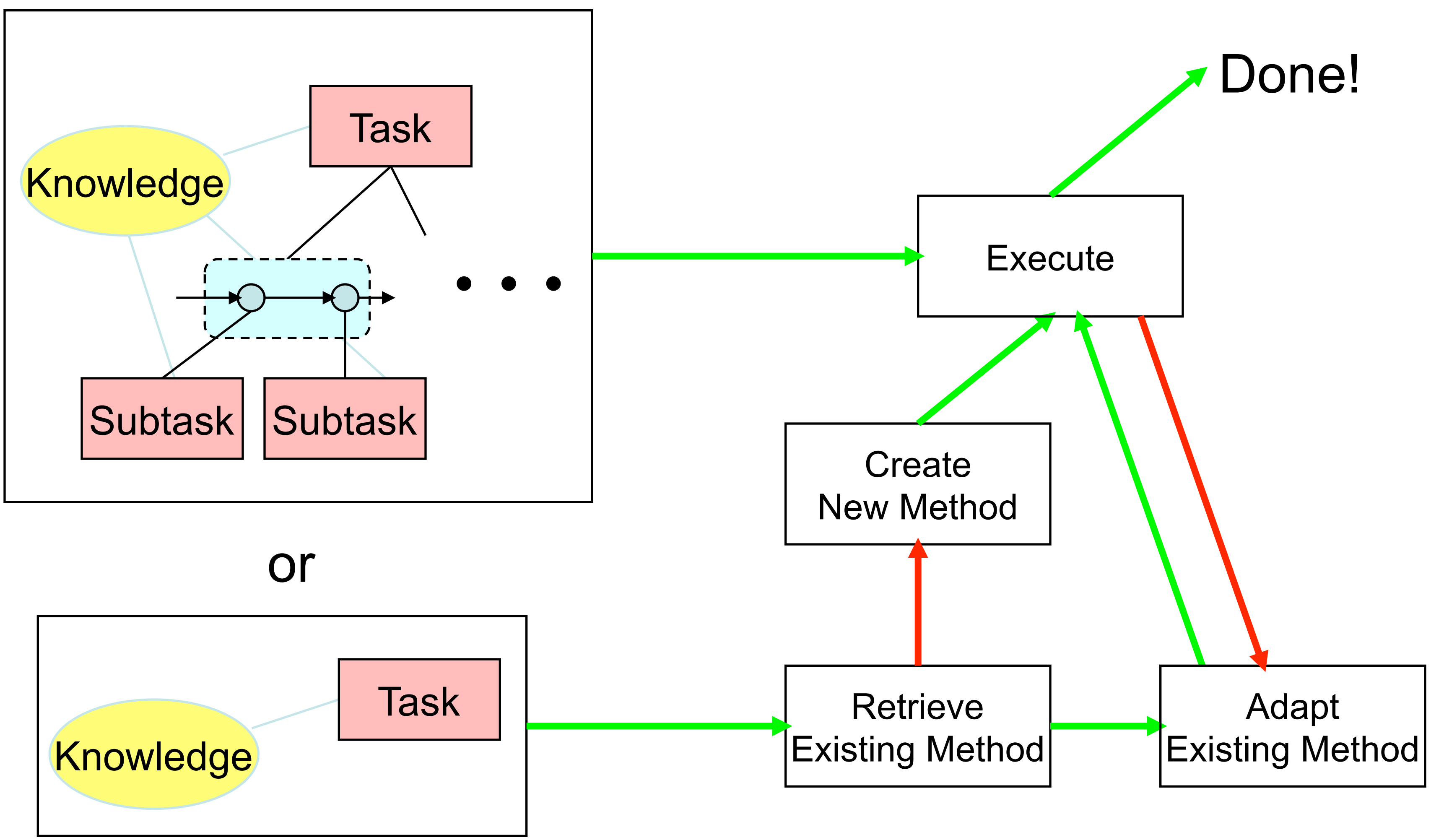
Knowledge in TMKL

- Foundation: LOOM
- Concepts, instances, relations
- Concepts and relations are instances and can have facts about them.
- Knowledge representation in TMKL involves LOOM + some TMKL specific reflective concepts and relations.

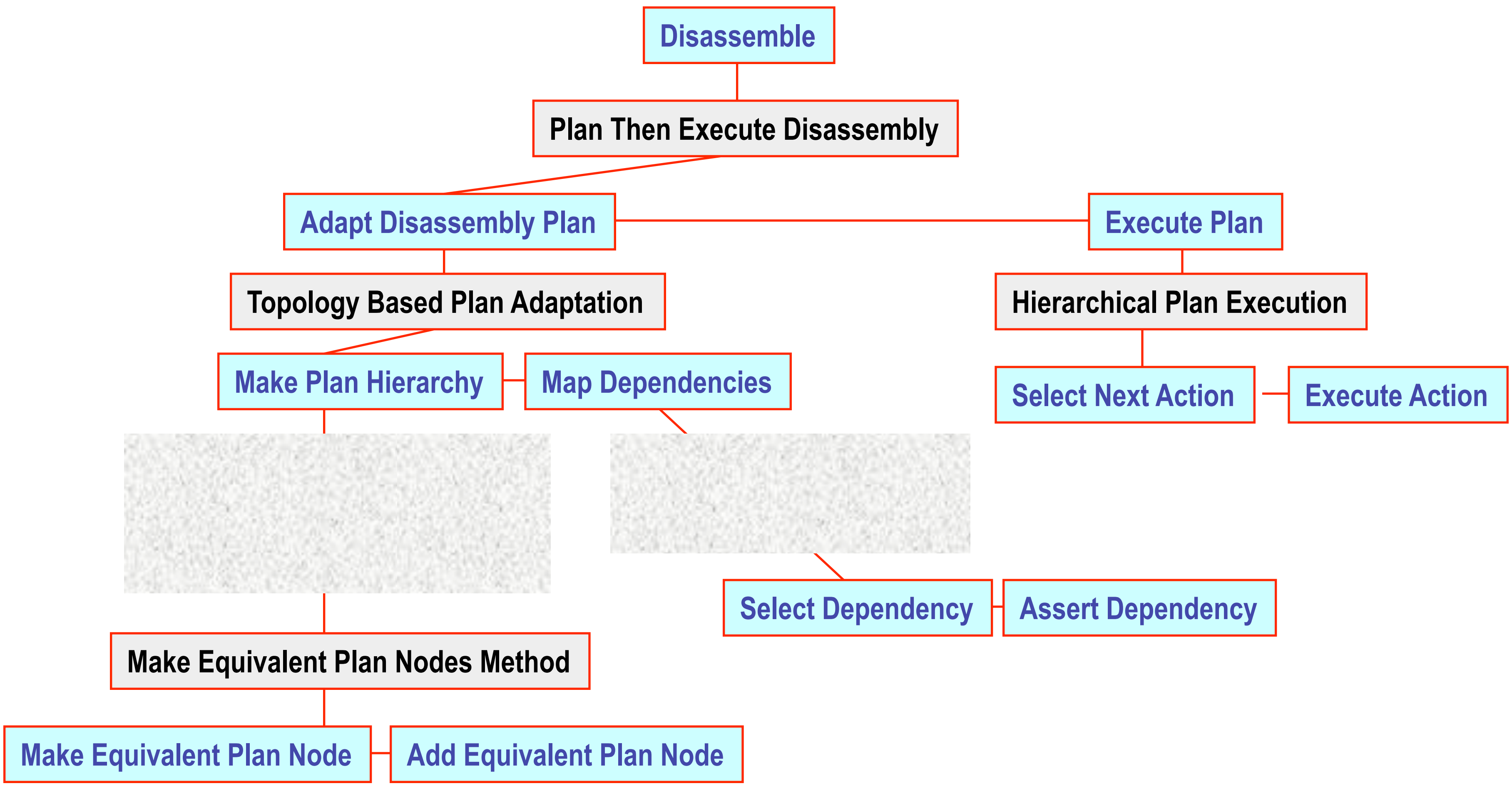
Sample Meta-Knowledge in TMKL

- generic relations
  - same-as
  - instance-of
  - is-a
  - inverse-of
- relation characteristics
  - single-valued/multiple-valued
  - symmetric, commutative
  - many more
- relations over relations
  - external/internal
  - state/definitional
- concepts of relations
  - binary-relation
  - unary-relation
  - same-as
  - is-a
  - inverse-of
- concepts relating to concepts
  - thing
  - concept
  - Meta-concept

REM's Functional Architecture



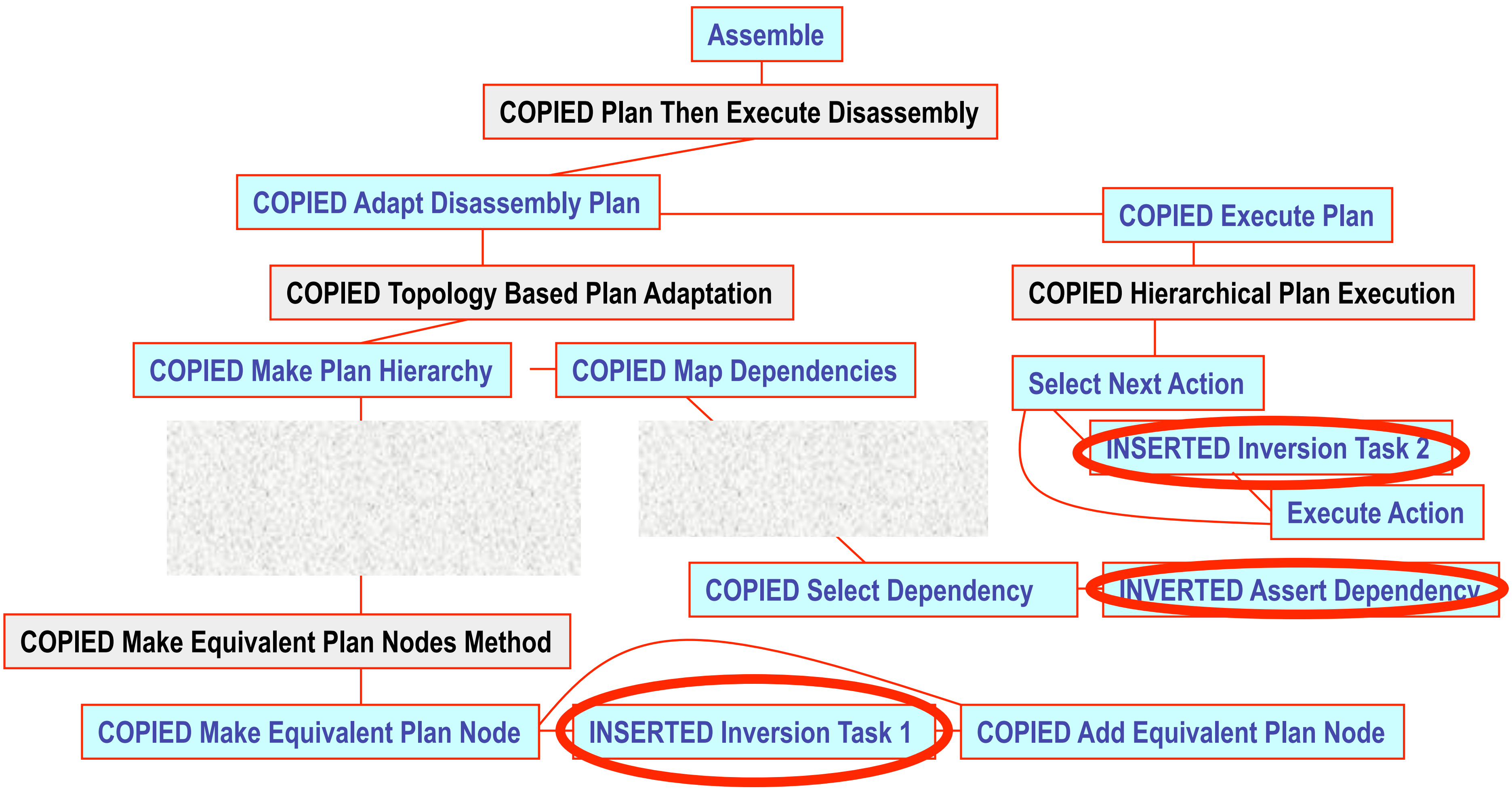
Pieces of ADDAM which are key to the Disassembly → Assembly Problem



Process for Addressing the Assemble Task by REM using ADDAM

- First the agent tries to find a method for the Assemble task. It doesn't have one.
- Next it tries to find a similar task which does have a method. It finds Disassemble.
  - The index is the input and output information provided in the task.
  - Similarity is determined by a combination of general rules plus domain-specific rules and assertions.
- Next it searches for a relation which links the effects of the two task. It finds Inverse-of.
- Finally, it uses this relation to modify components of the existing process to address the new process.

New Adapted Assembly Task



Changes to the TMK Model of ADDAM

- After the task which produces plan nodes: add a task which imposes the inverse-of relation on the type of the node.
  - e.g., Unscrew → Screw
- The (simple) task which asserts ordering dependencies is changed to assert the inverse-of ordering dependencies.
- After the task which extracts plan nodes from a plan: add a task which imposes the inverse-of relation on the type of the node.
  - e.g., Screw → Unscrew

Adaptation Strategy: Inversion

[Copy methods for known task to main task]

invertible-relations = [all relations for which inverse-of holds with some other relation]

invertible-concepts = [all concepts for which inverse-of holds with some other concept]

relevant-relations = invertible-relations + [all relations over invertible-concepts]

relevant-manipulable-relations = [relevant-relations which are internal state relations]

candidate-tasks = [all tasks which affect relevant-manipulable-relations]

FOR candidate-task IN candidate-tasks DO

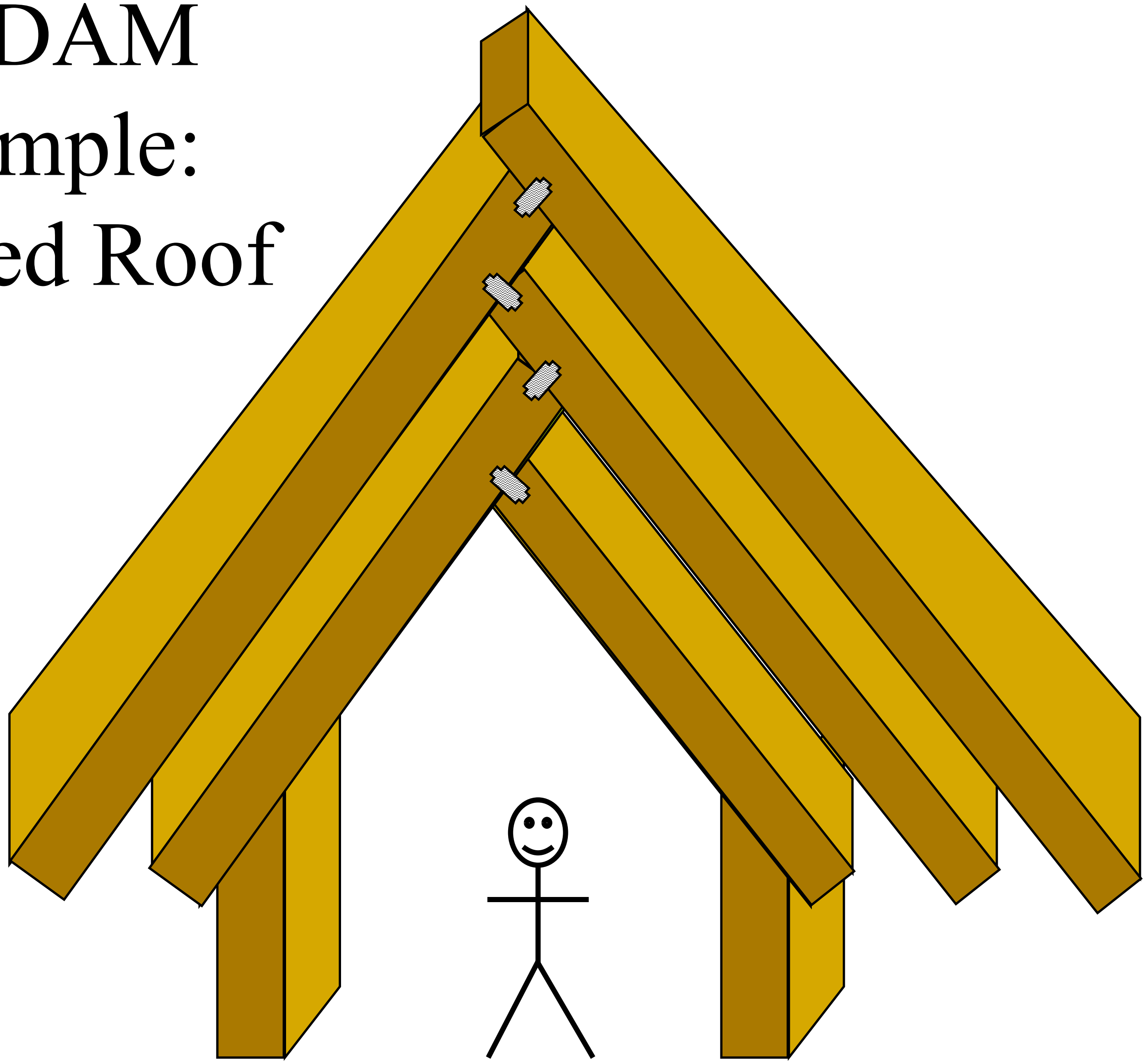
    IF [candidate-task directly asserts a relevant-manipulable-relations] THEN

        [invert the assertion for that candidate task]

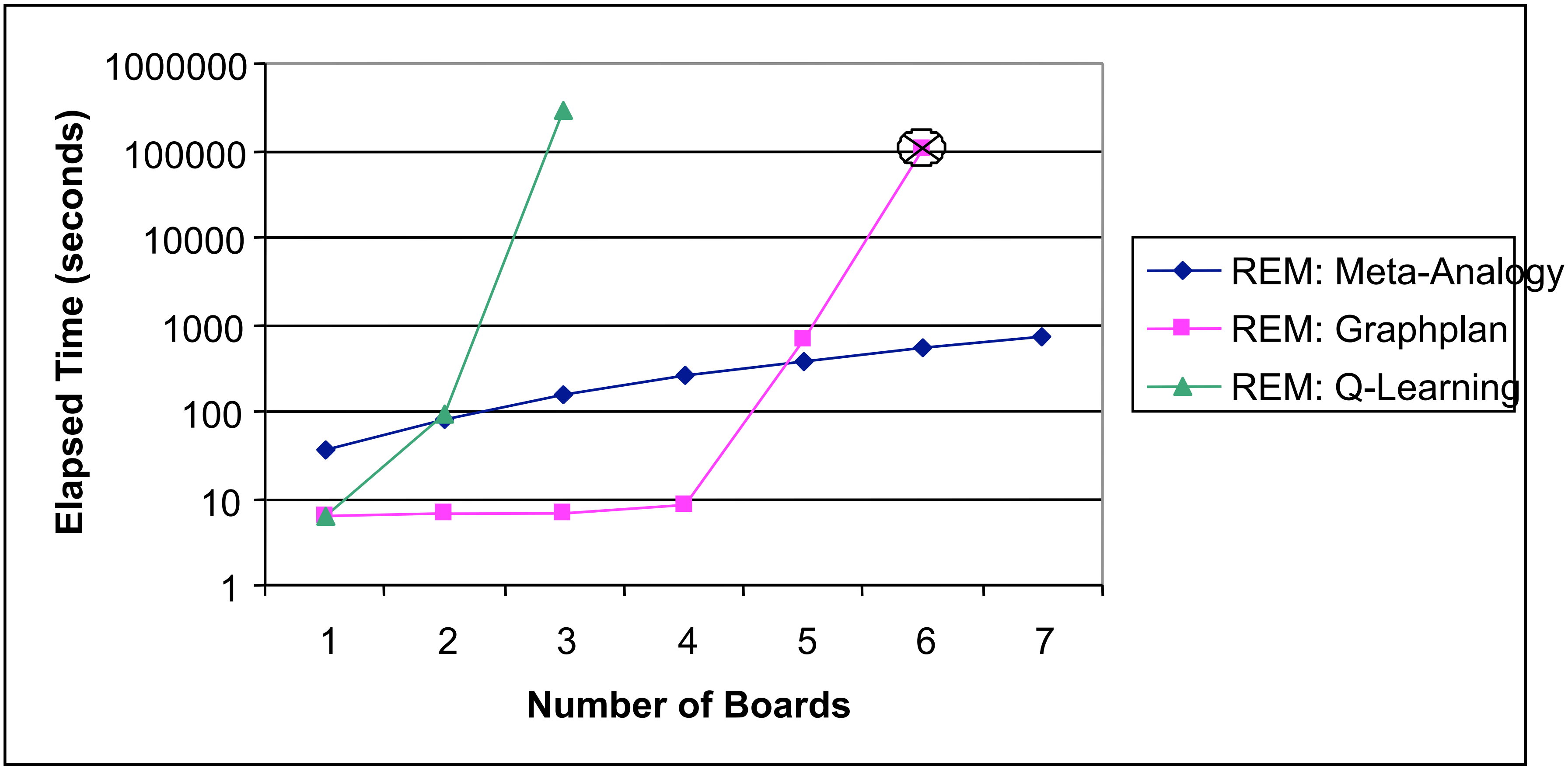
    ELSE IF [candidate-task produces an invertible output] THEN

        [insert an inversion task after candidate-task]

ADDAM Example: Layered Roof



Roof Assembly



What kinds of task differences can REM handle?

Inversion
Replication
Simple Generalization/Specialization

A limitation:

For complex problems, REM can only *localize* the needed modifications, not precisely identify them

An opportunity:

Use model-based meta-reasoning for localization;
Use local generative planning or reinforcement learning for identification.

Outline:

1. Introduction: model-based meta-reasoning for self-adaptation.

2. Proactive, goal-directed reconfiguration of reasoning processes.

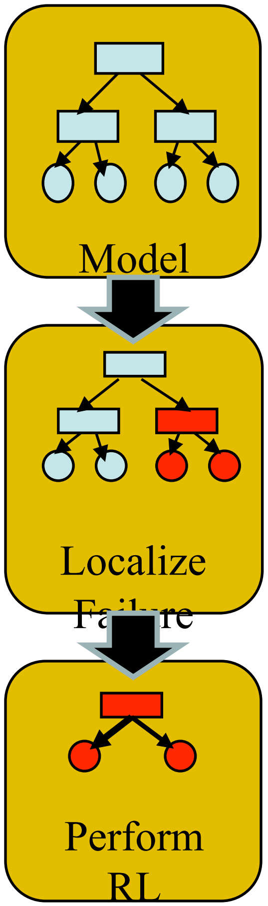
3. *Combining model-based meta-reasoning and reinforcement learning in retrospective failure-driven learning.*
   Jointly with Patrick Ulam, Joshua Jones & William Murdock

4. Retrospective, failure-driven repair of domain knowledge.

5. Conclusions: model-based meta-reasoning for self-adaptation.

Using Model-Based Meta-Reasoning for Localizing Reinforcement Learning

- Endow the agent with a TMK model of its own design.
- Upon failure, use model-based meta-reasoning to localize the failure to a specific element of task execution
- Use reinforcement learning (RL) to adapt agent’s reasoning at identified location



Domain: Interactive Games

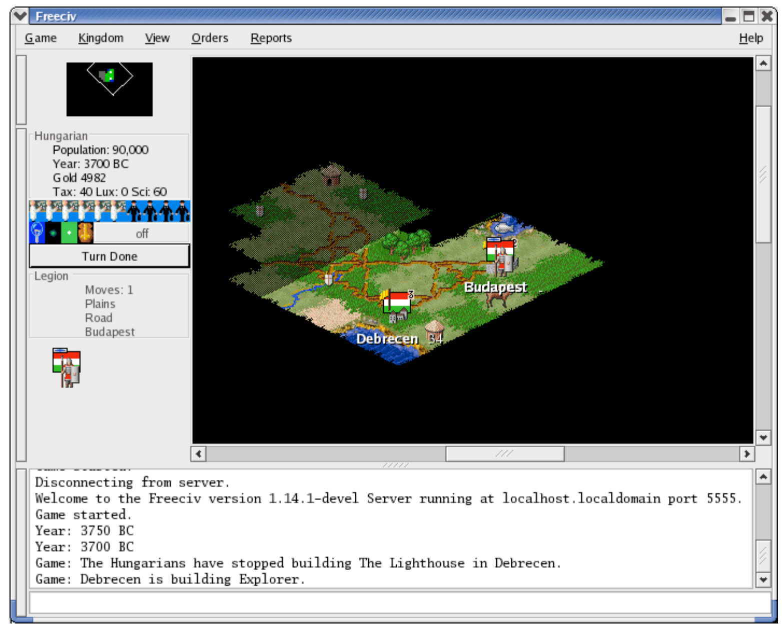
Freeciv (www.freeciv.org) is a popular, interactive turn-based strategy game.

A human plays the game against multiple software agents.

The goal is to conquer the world.

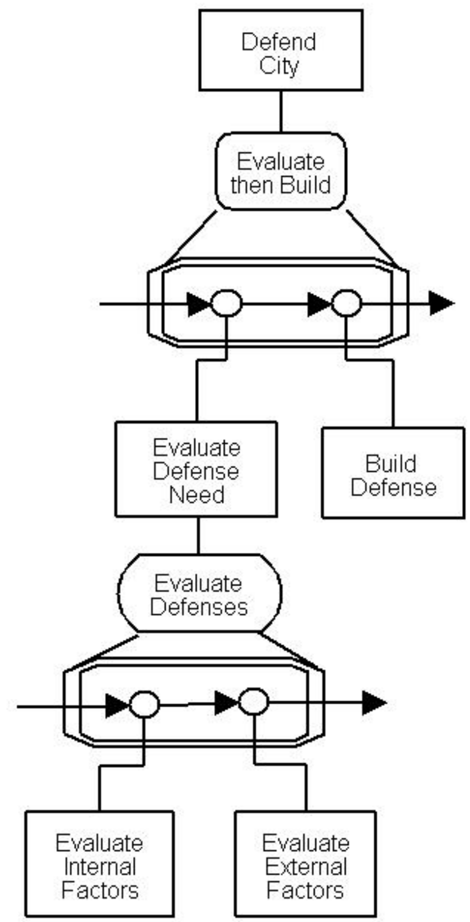
Defend The City Task

- Goal
  - Defend the starting city from enemy civilizations for as long as possible
- Possible actions
  - Building the unit with highest defensive value
  - Producing wealth
- Success conditions
  - Survive 100 turns
- Failure conditions
  - City is defeated
  - City revolts



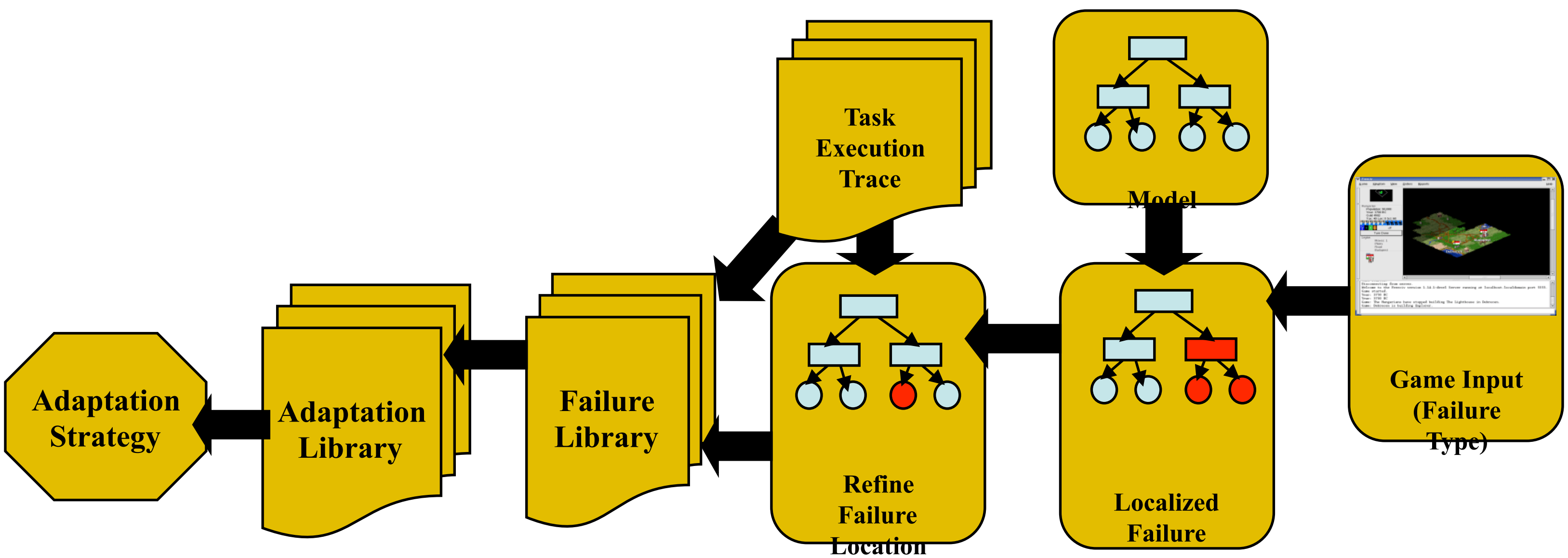
Defend the city model

- Tasks
  - Defend City
  - Evaluate Defense Needs
  - Build Defenses (Procedure)
  - Evaluate Internal Factors (Procedure)
  - Evaluate External Factors (Procedure)
- Methods
  - Evaluate and Build
  - Evaluate Defenses
- Knowledge
  - Different for each task
  - E.g. evaluate external factors produces knowledge about the number of enemy units nearby



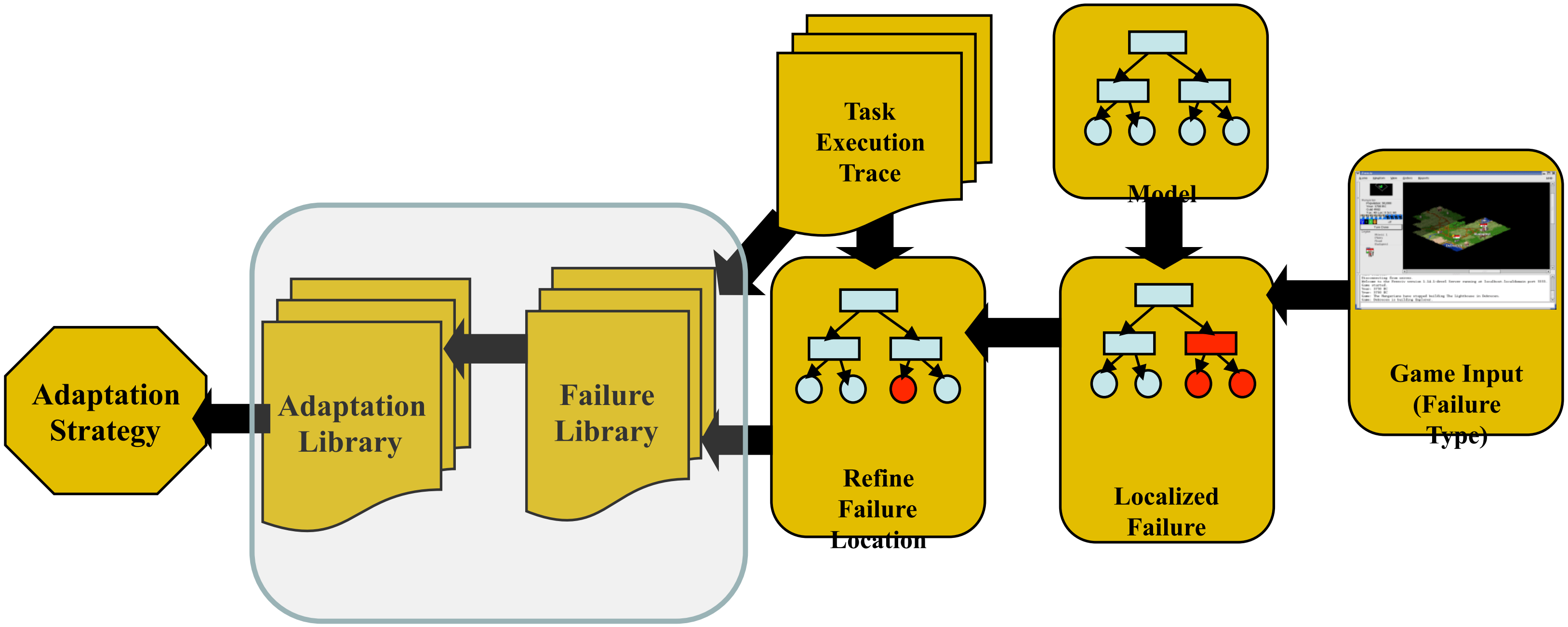
Adapting the Defend City Task by Model-Based Meta-Reasoning

- Upon task failure, failure type used to localize failure within model
- Execution trace used to further narrow space of possible failures locations
- Adaptation for each type of failure provided via user-suppplied adaptation library
- Adaptations consist of small changes to procedural tasks (leaf nodes)



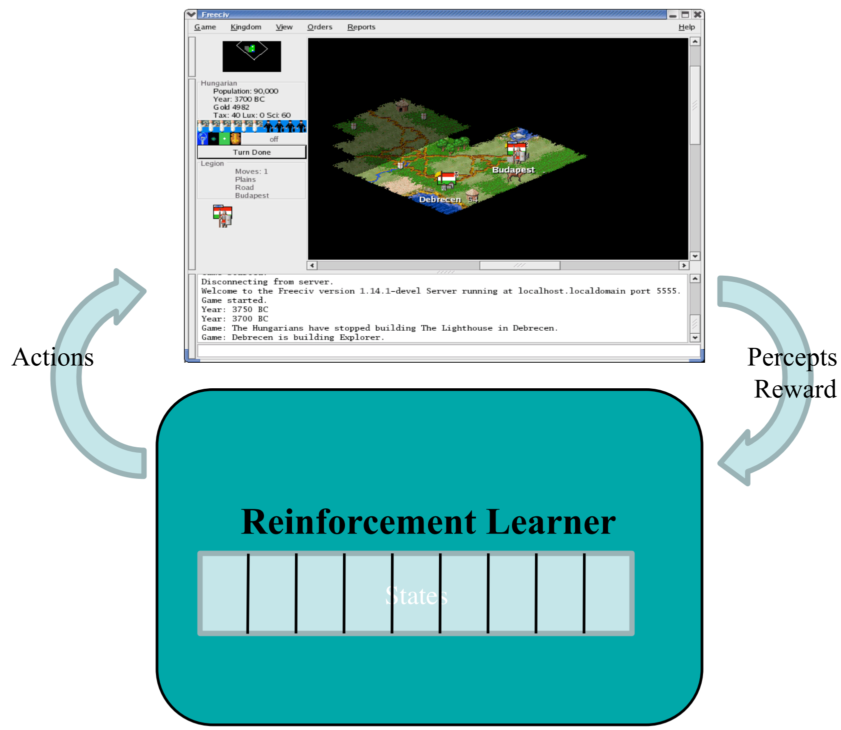
Adapting The Defend City Task Via Pure Model-Based Reasoning

- Space of possible adaptations and failures is large
- Requires significant knowledge engineering to make the failure and adaptation library
- Difficult to determine if adaptation library is insufficient



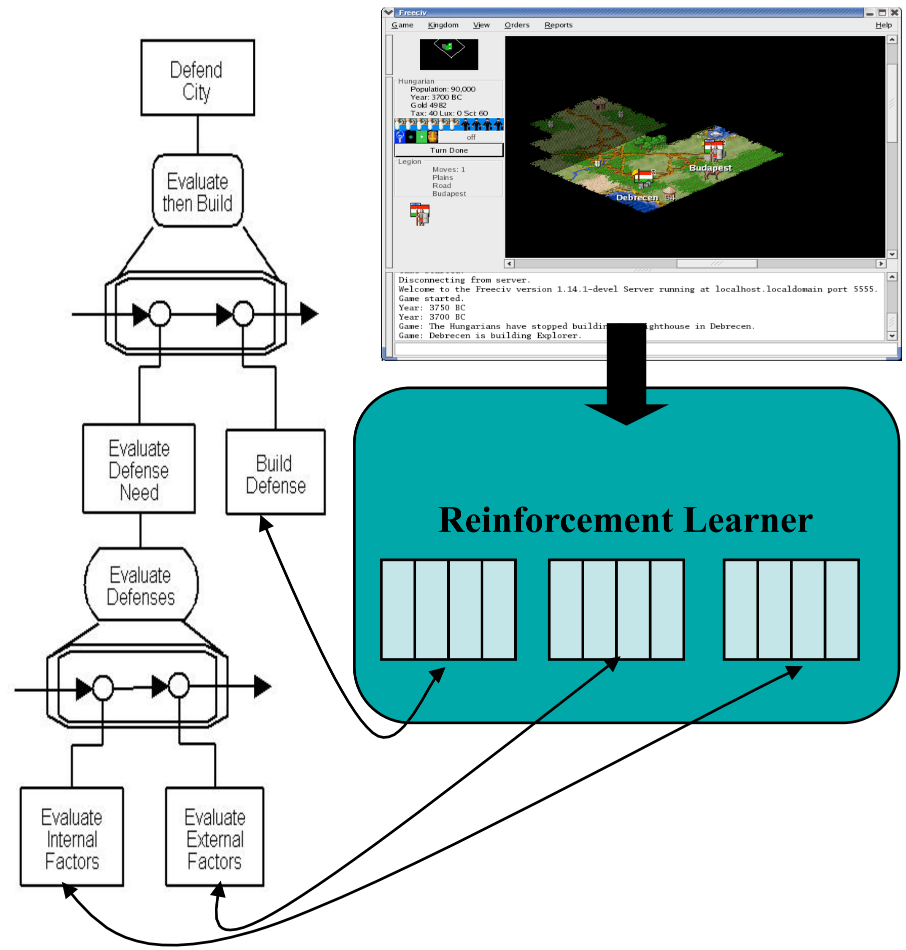
Adapting the Defend City Task by Reinforcement Learning

- State space consisting of 9 binary variables
  - E.g. Are there are less then X enemy units near the city?
  - E.g. Are there are less then Y defensive units currently stationed at the city
- Two actions
  - Build defensive unit
  - Build wealth
- Negative reward signal received upon failure



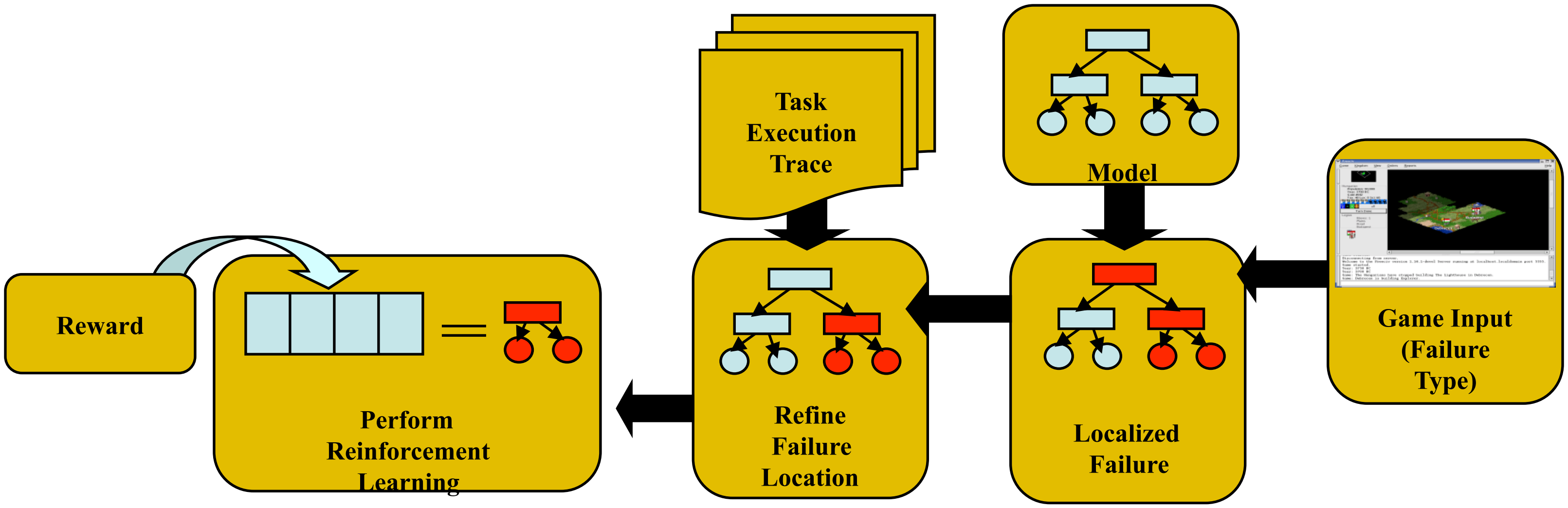
Combining Model-Based Meta-Reasoning and Reinforcement Learning

- Use model as guide for divide state space
- Associate each subdivision with specific portion of model
- Each small reinforcement learner can receive separate reward signal



Adaptation in the Hybrid Technique

- Upon task failure, failure type used to localize failure within model
- Execution trace used to narrow space of possible failures locations
- Only portions of model identified receive reward signal



Ulam, Jones, Goel & Murdock 2005, 2008

Evaluation

- Experimental Setup
  - Performed 100 trials of 100 turns for each agent on smallest map setting
  - Each trial for each agent shared same map
  - 8 Built-in AI opponents at hardest difficulty
  - Agent limited to a single city

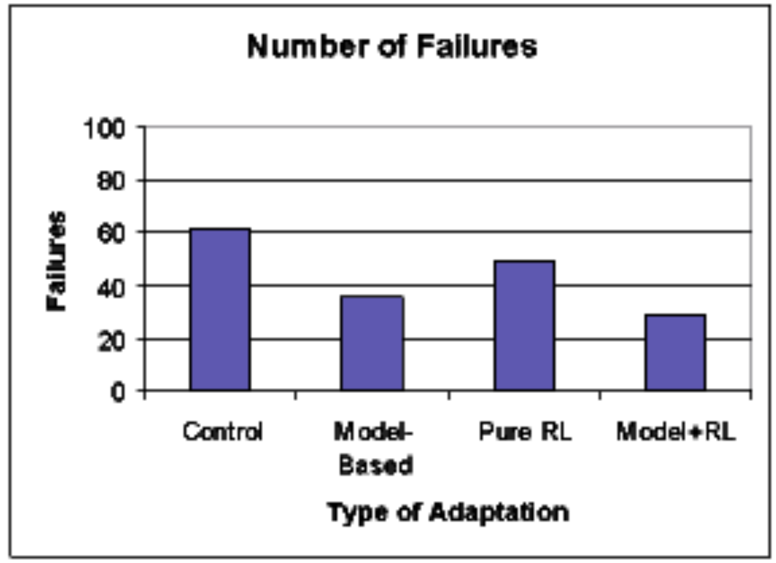
- Evaluation Metrics
  - Number of failures
  - Mean time between failures
  - Number of attacks successfully defended

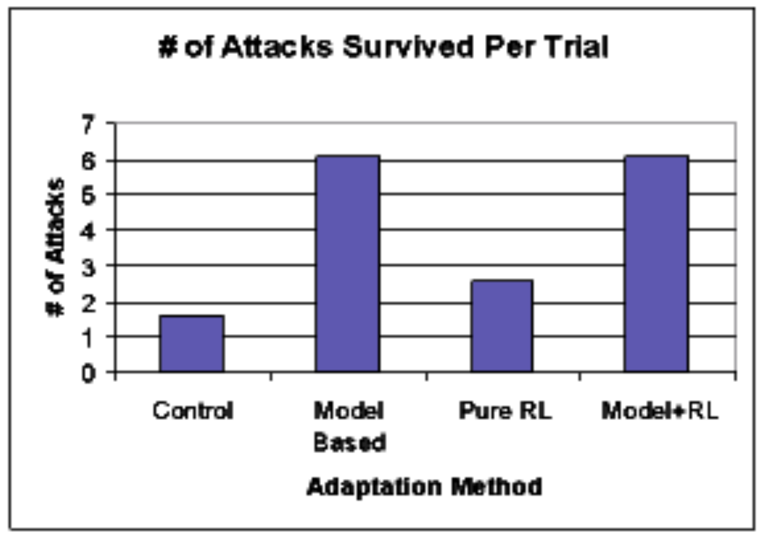
Experimental Agents

- Control
  - Agent attempts to maintain 1 defensive unit, no adaptation
- Pure model-based reflection agent
  - Starts from control
  - Adapts via user defined adaptation library
- Pure reinforcement learning agent
  - Initialized to always build wealth
  - Q-Learning
- Hybrid agent
  - Initialized to always build wealth

Experimental Results

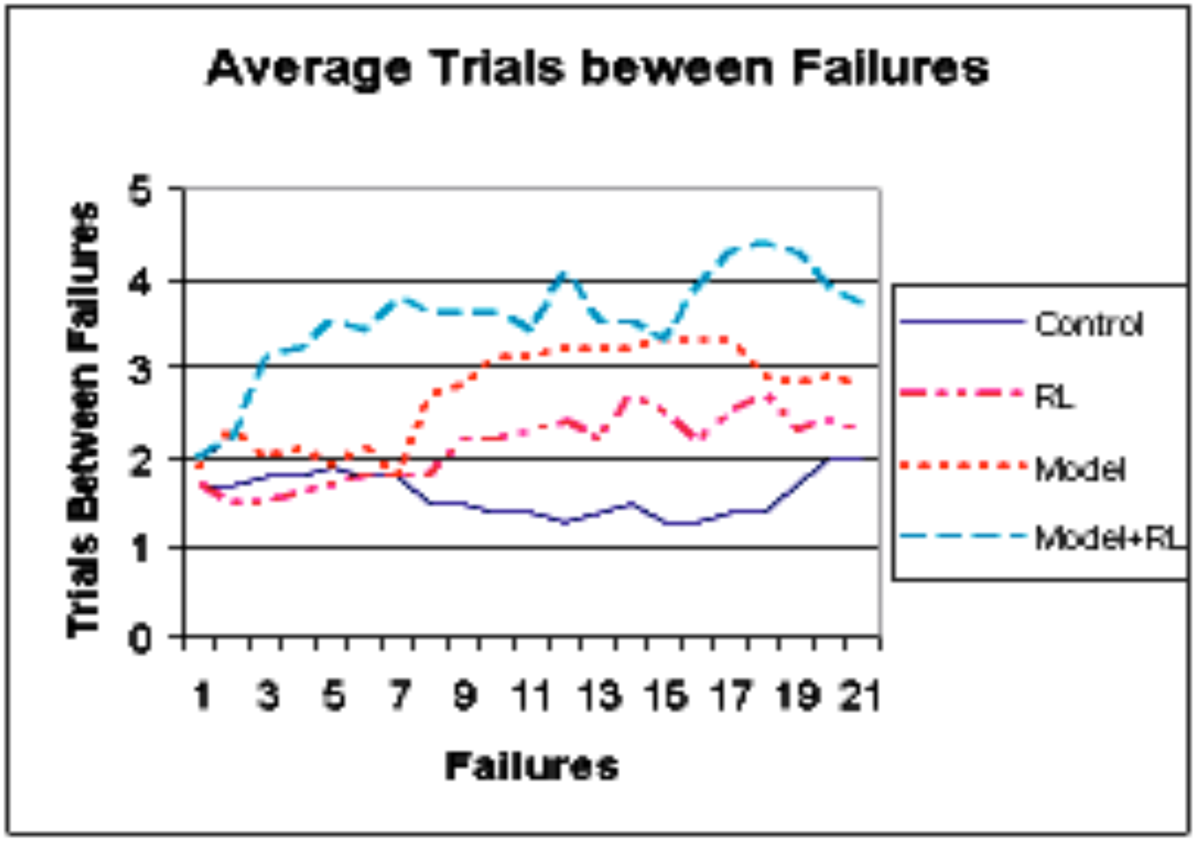
- Number of trials failed directly measures success of agent
  - Less failures indicates better performance
- Number of attacks per trial
  - More attacks survived indicates higher performance
- Hybrid model-based/RL method combines low failure rate with good survival rate.





Experimental Results

- Average time between failures
  - Assumes the better the agent learns the task, the longer the period between failures
  - Rate of increase indicator of speed of learning



Outline:

1. Introduction: model-based meta-reasoning for self-adaptation.

2. Proactive, goal-directed reconfiguration of reasoning processes.

3. Combining model-based meta-reasoning and reinforcement learning.

4. *Retrospective, failure-driven repair of domain knowledge.*
   Joint Work with Joshua Jones.

5. Conclusions: model-based meta-reasoning for self-adaptation.

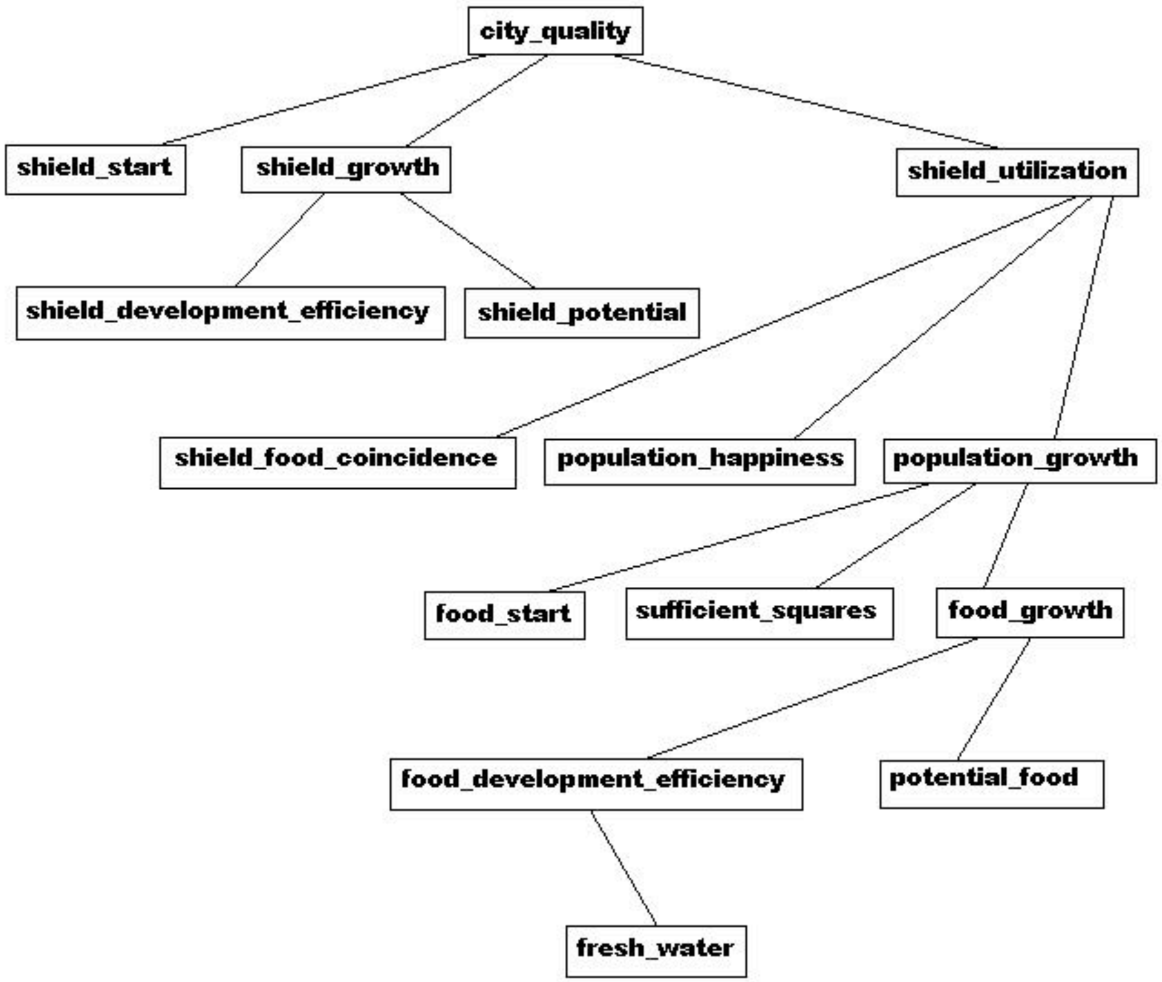
Predictive Knowledge

- Metareasoning techniques for adaptation of agent processes in REM use predictive knowledge expressed in a functional self-model.
- This predictive knowledge enables the agent to both detect its failures and diagnose the causes.

Adaptation of Domain Knowledge

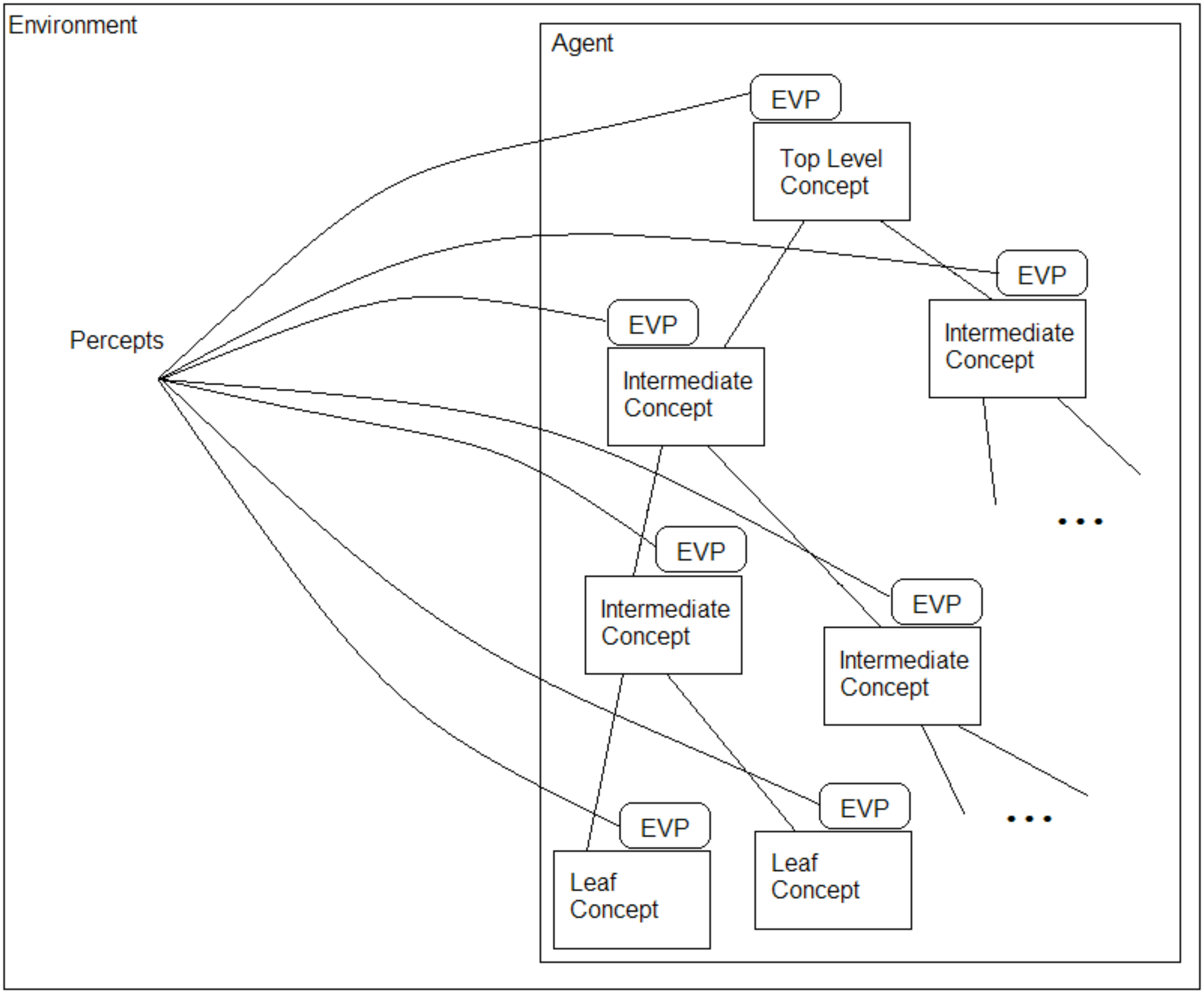
- How can a metareasoning process be used to adapt an agent's domain knowledge?
- We take an analogous approach, providing the agent with knowledge representations that explicitly represent predictive implications of conceptual knowledge.
- We call the metaknowledge associated with each concept an empirical verification procedure (EVP).

Classification Knowledge for Locating Cities in Freeciv



Abstraction Networks

- Abstraction networks (ANs) is a representation that adds EVP metaknowledge to concepts in a classification hierarchy.
- ANs are intended to be as general as possible within hierarchical classification, not committing to:
  - A particular type of sub-learner
  - A specific diagnostic probe-selection method



The AN representation, instantiating EVP theory in the context of compositional classification.

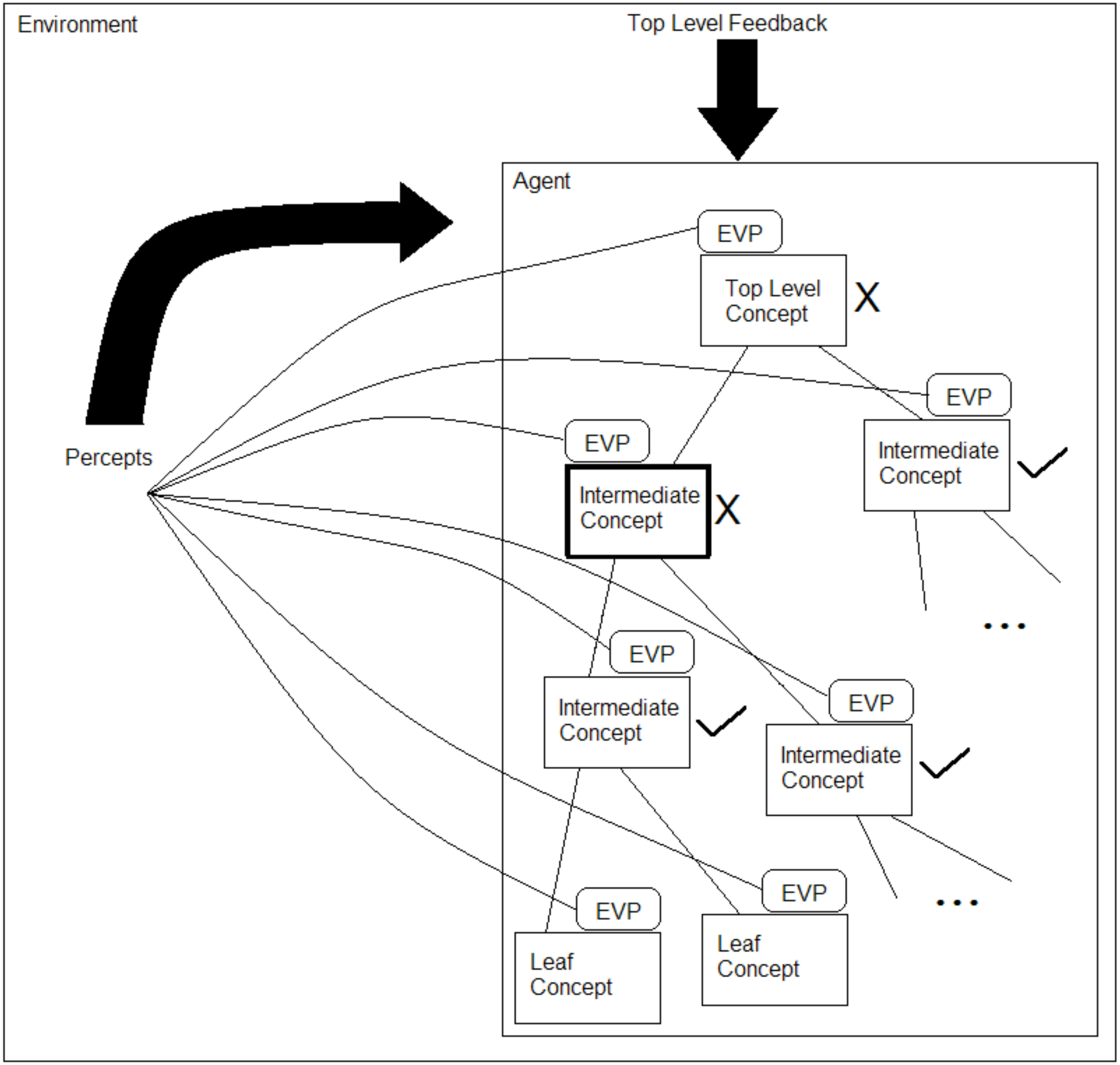


Illustration of non-exhaustive diagnosis in an AN.

Auger: Basic Hypotheses

- Faster learning with ANs (trees + EVPs)
- Faster learning even if imperfect knowledge engineering (imperfect trees + EVPs)
- Faster learning even if absent EVPs
- Refinement of concept semantics (bin size)

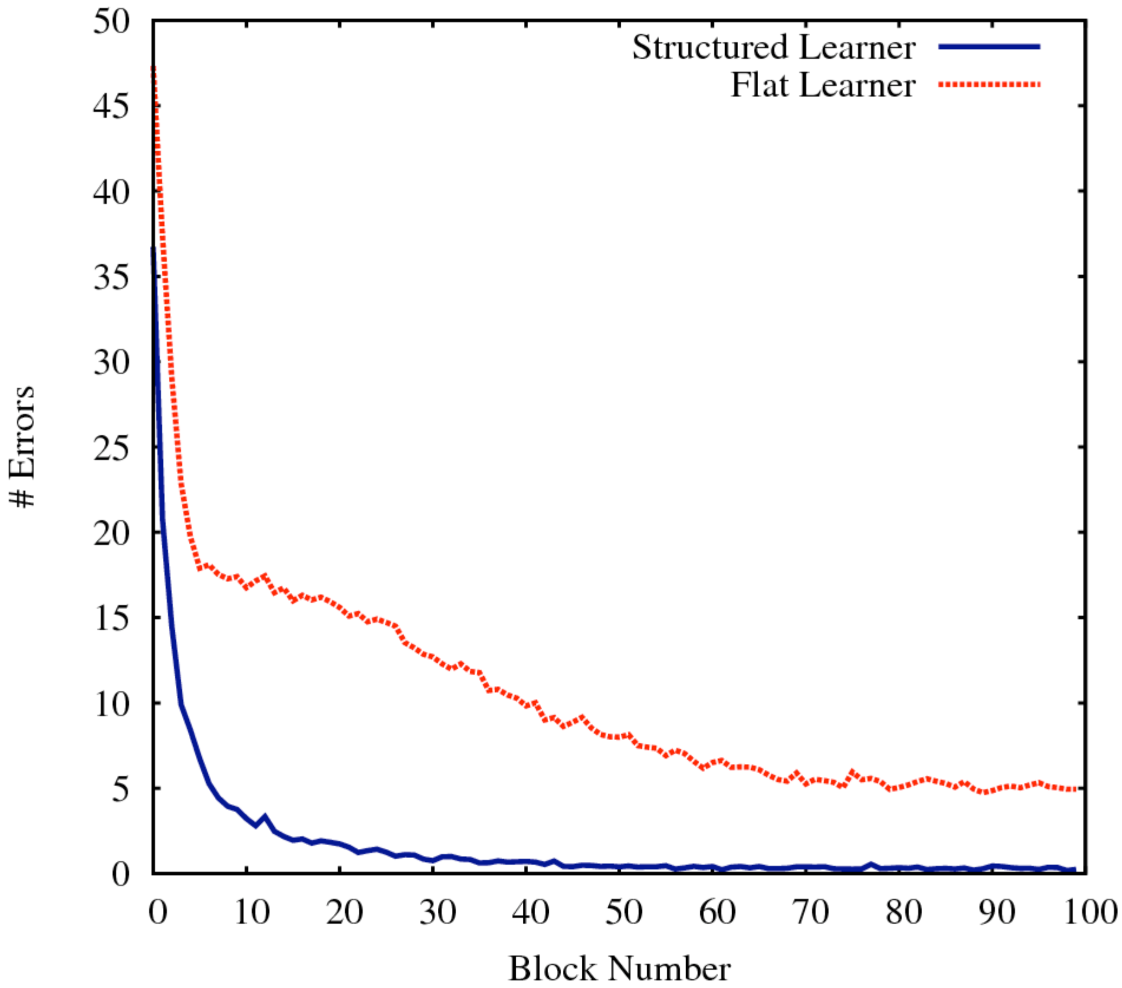
Auger: Basic Experiments

- We have performed basic experiments in 4 domains
  - Synthetic
  - FreeCiv Game Playing
  - DJIA prediction
  - NFL prediction
- 3 learner types have been used
  - Table-based rote learners
  - ANNs
  - kNNs

Auger: Basic Procedures

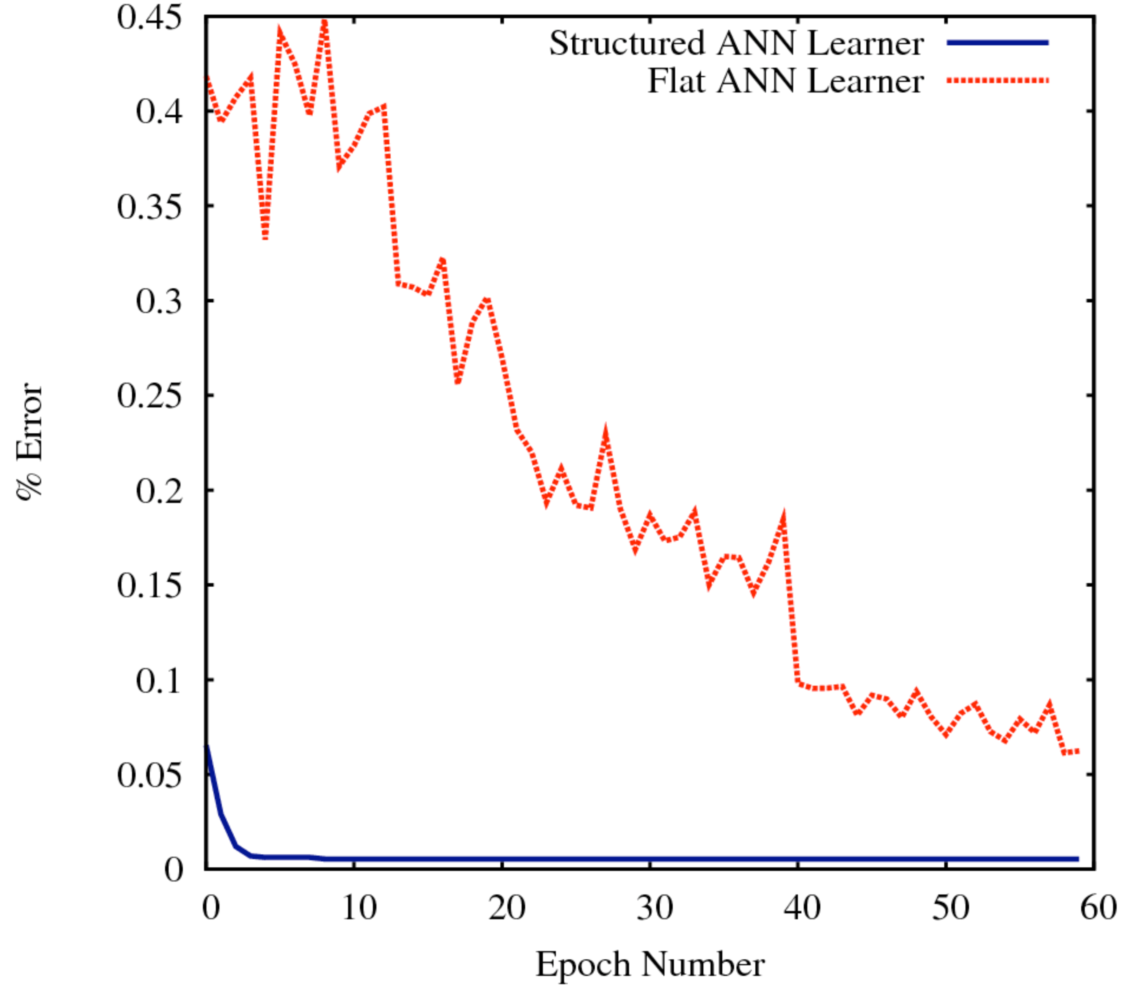
- Two kinds of diagnosis
  - Non-exhaustive “causal backtracing”
  - Full EVP execution
- Two kinds of training
  - On-line, sequential
    - Sequence of examples split into equal sized blocks
    - For each example, perform inference and check for error
    - Then, train
  - Batch
    - Used with ANNs only
    - Typical training set/test set approach with epochs

Results: Synthetic Domain, Rote Learners



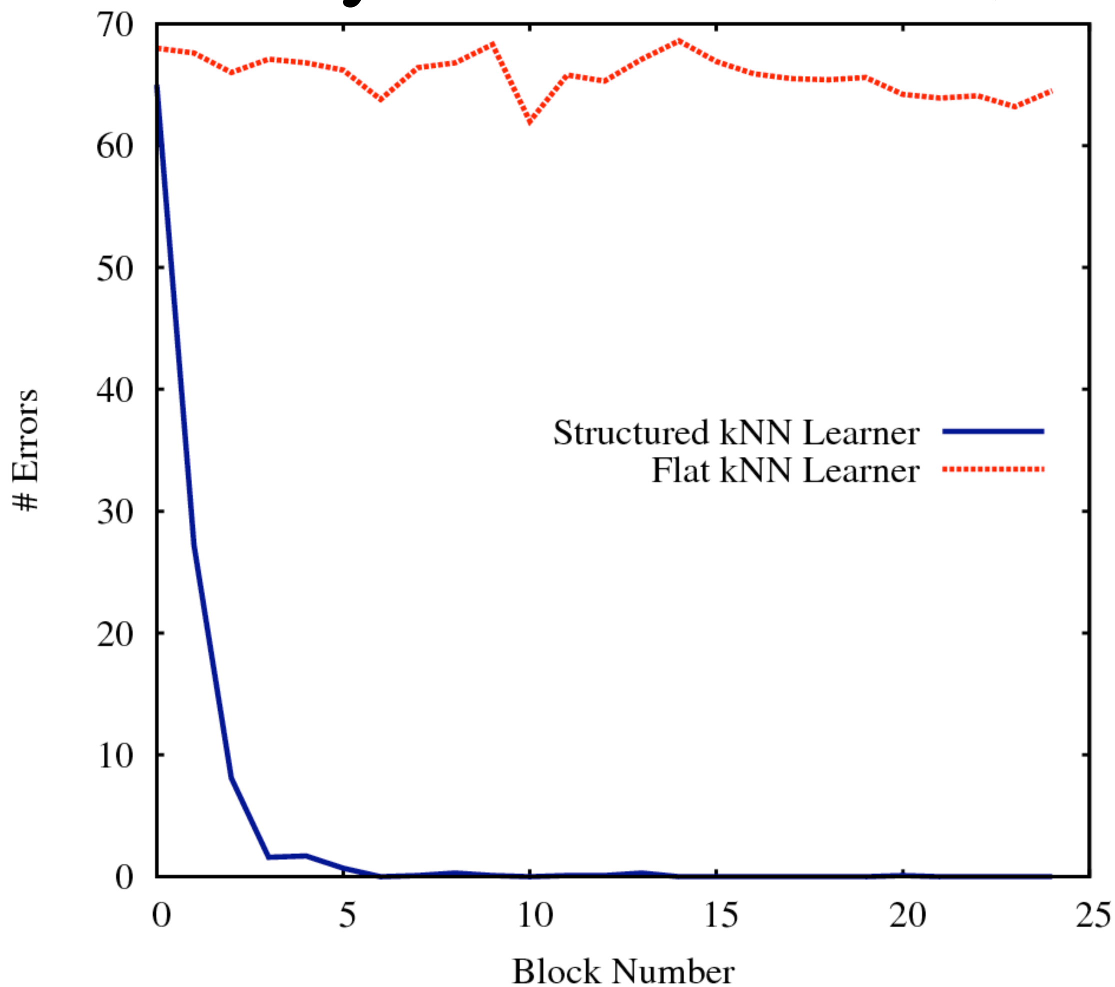
Layer sizes 16-8-4-2-1, 3 choices per node, block size 100 examples, average of 100 trials.

Results: Synthetic Domain, ANNs



Layer sizes 16-8-4-2-1, 3 choices per node, training set size 1000, test set size 1000, average of 5 trials, full EVP evaluation.

Results: Synthetic Domain, kNNs



Layer sizes 16-8-4-2-1, 4 choices per node, block size 100 examples, average of 10 trials, k=1.

FreeCiv Results

	AN learner	Flat Table Learner	
	7^{th} block	7^{th} block	21^{st} block
Without city improvements	24%	(4%)	1%
With city improvements	29%	7%	10%

Performance vs. Unstructured Learners

Conclusions

Why meta-reasoning? Two more reasons: proactive, goal-directed self-adaptation of reasoning processes, and localization of situated learning.

How meta-reasoning? Model-based meta-reasoning. Models that are compositional and that encode predictive knowledge. Models that describe the design of the agent.

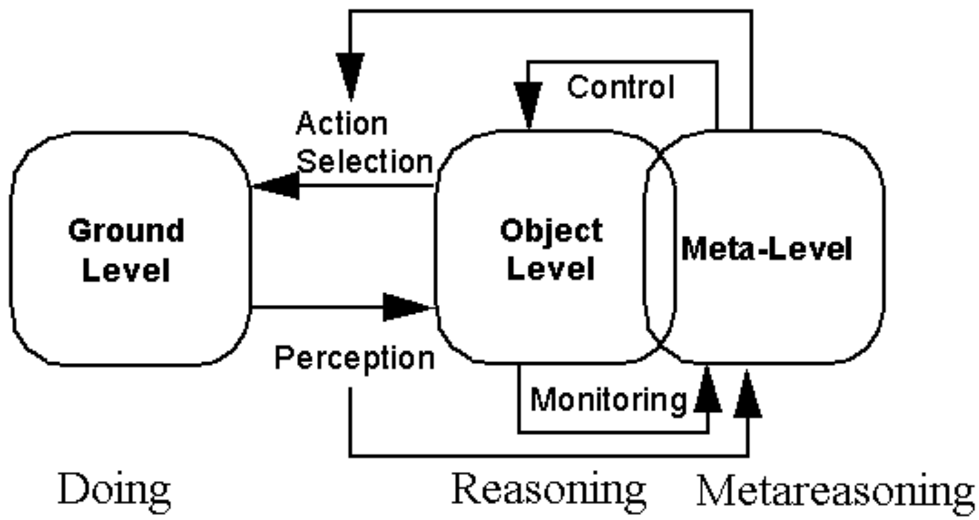
Conclusions

- In proactive reconfiguration of reasoning processes, representation of the agent's design, function, and teleology (in the form of TMK models) enables meta-analogies.

- In retrospective repair of domain knowledge, representation of empirical verification procedures (EVPs) in abstraction networks (ANs) enables learning of content of domain knowledge.

Implications for Metareasoning

- Our work suggests an elaboration of the basic metareasoning architecture:



Acknowledgements

Eleni Stroulia, University of Alberta

Retrospective, Failure-Driven Self-Adaptation
Self-Adaptation in Reactive Agents

J. William Murdock, IBM TJ Watson Center

Proactive, Goal-Directed Self-Adaptation (REM)
Task-Method-Knowledge Language (TMKL)

Acknowledgements

Joshua Jones, University of Maryland BC

Self-Adaptation of Domain Knowledge (Augur)

Spencer Rugaber, Georgia Tech

Self-Adaptation in Game Playing Agents (GAIA)

DARPA Evolutionary of Design of Complex Systems
NSF Science of Design