Learning by Observation in Complex Domains

Michael van Lent and John Laird
Artificial Intelligence Lab
University of Michigan
1101 Beal Ave.
Ann Arbor, MI 48109
{vanlent,laird}@umich.edu

Intelligent Agent Bottleneck

- High-fidelity behavior requires lots of knowledge
  - 5,200 rules for TacAir-Soar = medium-fidelity
- Building knowledge-rich agents is very costly
  - > 18 person years for TacAir-Soar
- Where does all the time go?
  - Knowledge Acquisition, design, implementation, testing, debugging, extension, refinement, redesign, …
- Solution: Automatic Knowledge Generation

Problem Statement

- Develop task performance agents in complex domains
- Transfer expert's performance knowledge to agent.
  - Expert doesn't communicate knowledge or learn tools
  - Programmer doesn't become expert
  - Generate performance knowledge that matches expert
- Solution: Learning by Observation
  - Expert just performs the task
  - Programmer only learns a few details
  - Knowledge is based on expert's behavior

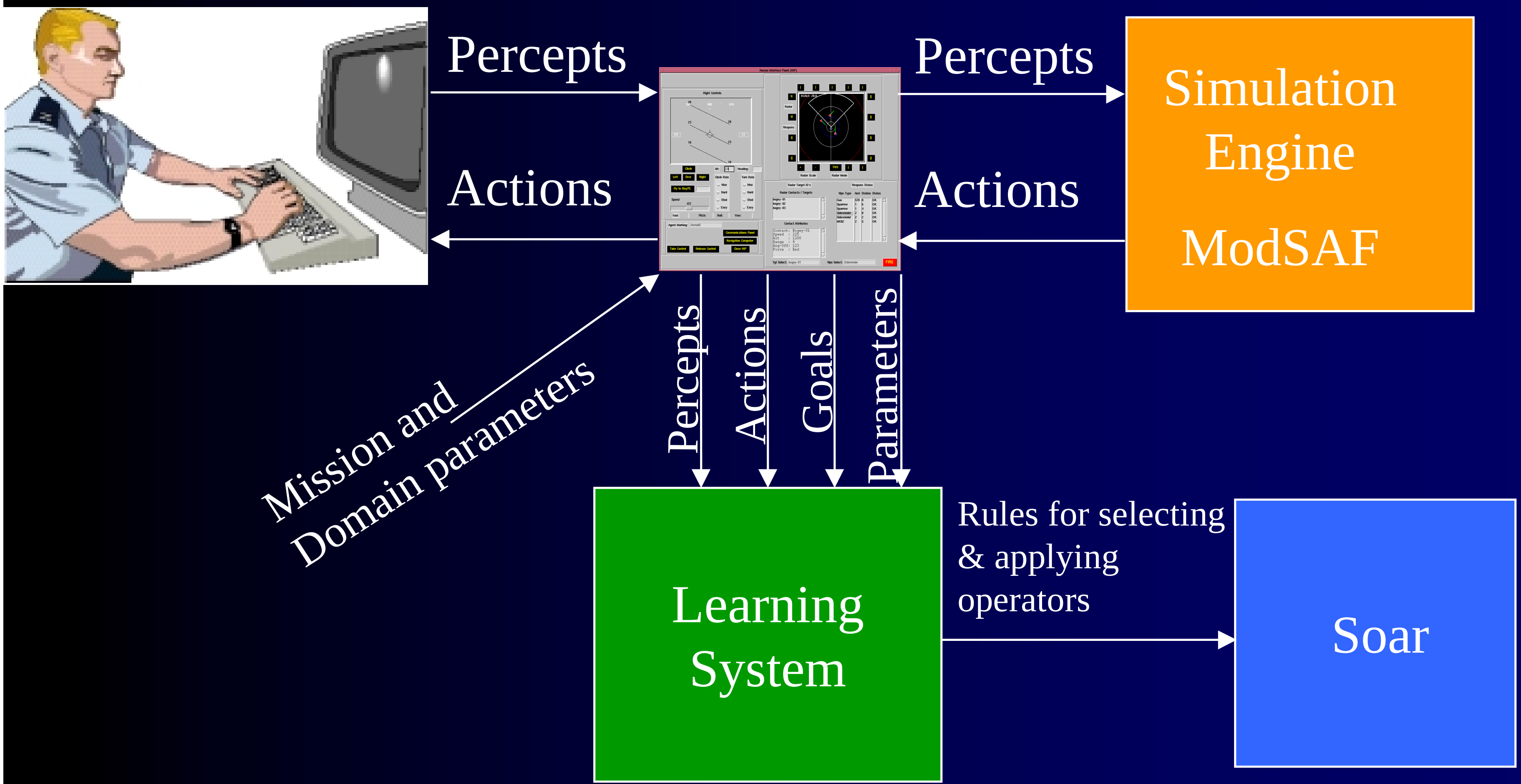
Approach

- Capture multiple traces of human behavior
  – Sensory data, active goals, actions
- Induce underlying knowledge
  – Rules for selecting & applying hierarchical operators
- Built on ideas from Behavior Cloning (Sammet et al.)
  – Add annotation of current operators/goals
  – Include more domain and mission information
  – Generate more complex and flexible execution structures

What is learned?

- Operator proposal productions
  - LHS: external sensors, internal features, operator hierarchy
  - RHS: operator proposal
- Operator application productions
  - LHS: external sensors
  - RHS: external action
- Goal achieved productions
  - LHS: external sensors, internal features, operator hierarchy
  - RHS: creates internal features (<OP>-goal-achieved *YES*)
  - Persistent and non-persistent features
    - Use I-support and O-support
    - Learn operators to remove persistent features

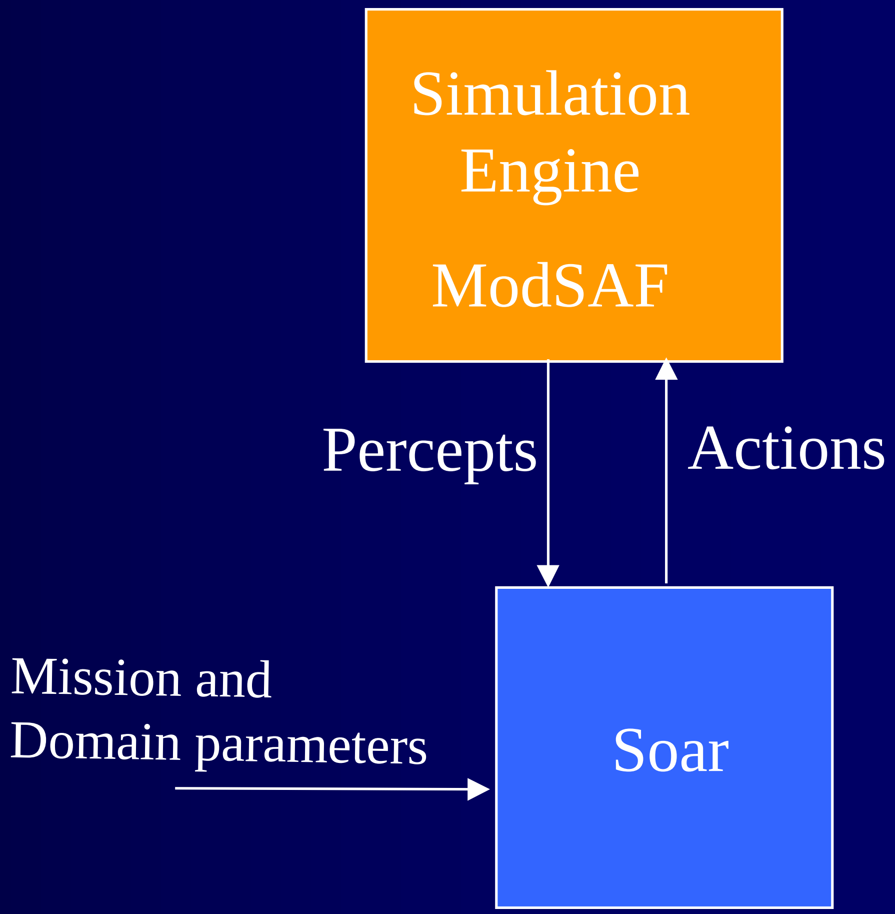
KnoMic System Structure



How are Operators Learned?

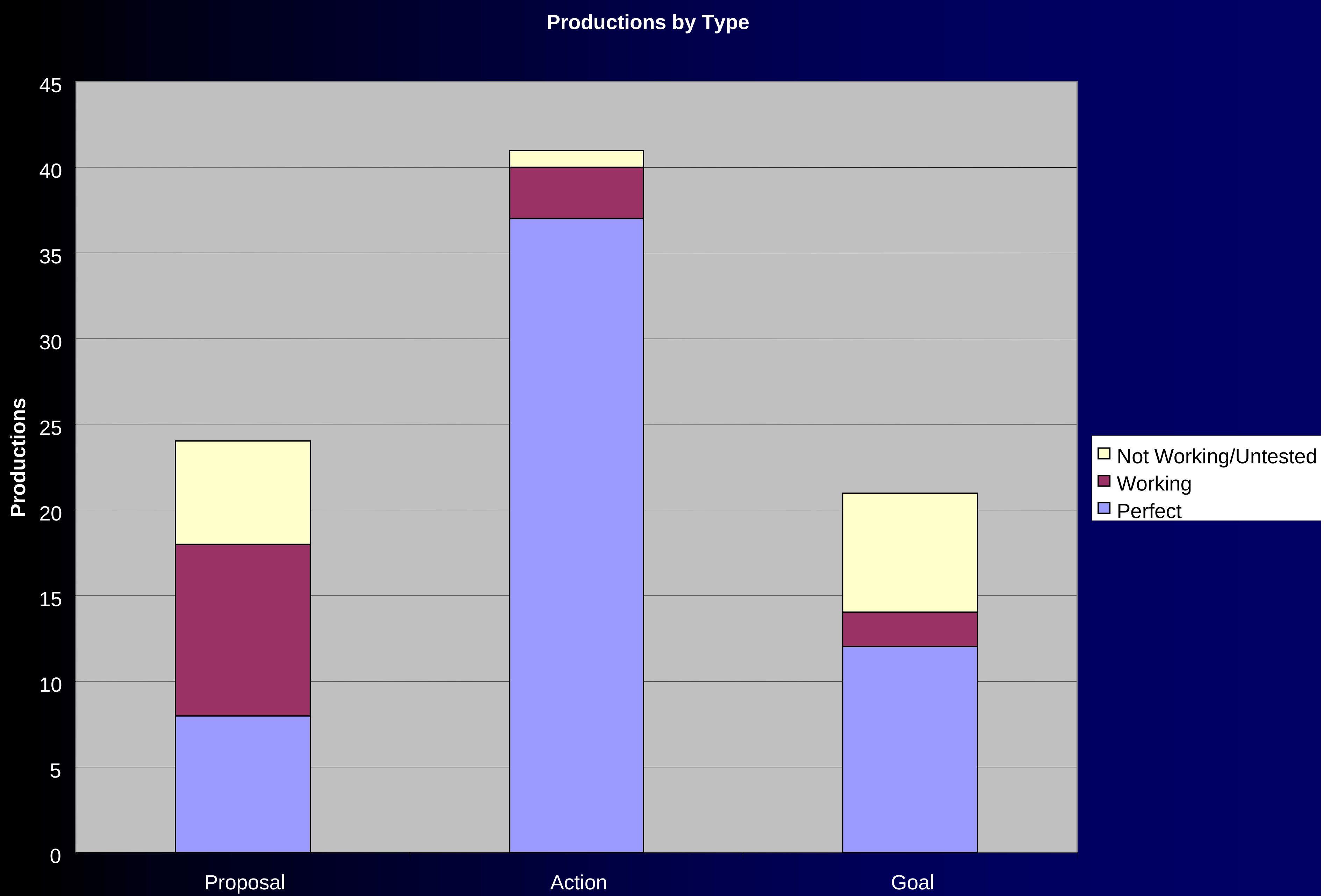
- Specific to General learning algorithm
- Pre-conditions and Action conditions
  - First operator selection: Everything is a pre-conditions
  - Subsequent selections: Remove anything not matched
  - Result: Most specific set of conditions true at every selection
- Goal conditions
  - First operator termination: Everything that changed recently
    - Recent Changes Heuristic
  - Subsequent terminations: Remove anything that didn’t change
  - Result: Most specific set of conditions that changed just before every termination

KnoMic System Structure

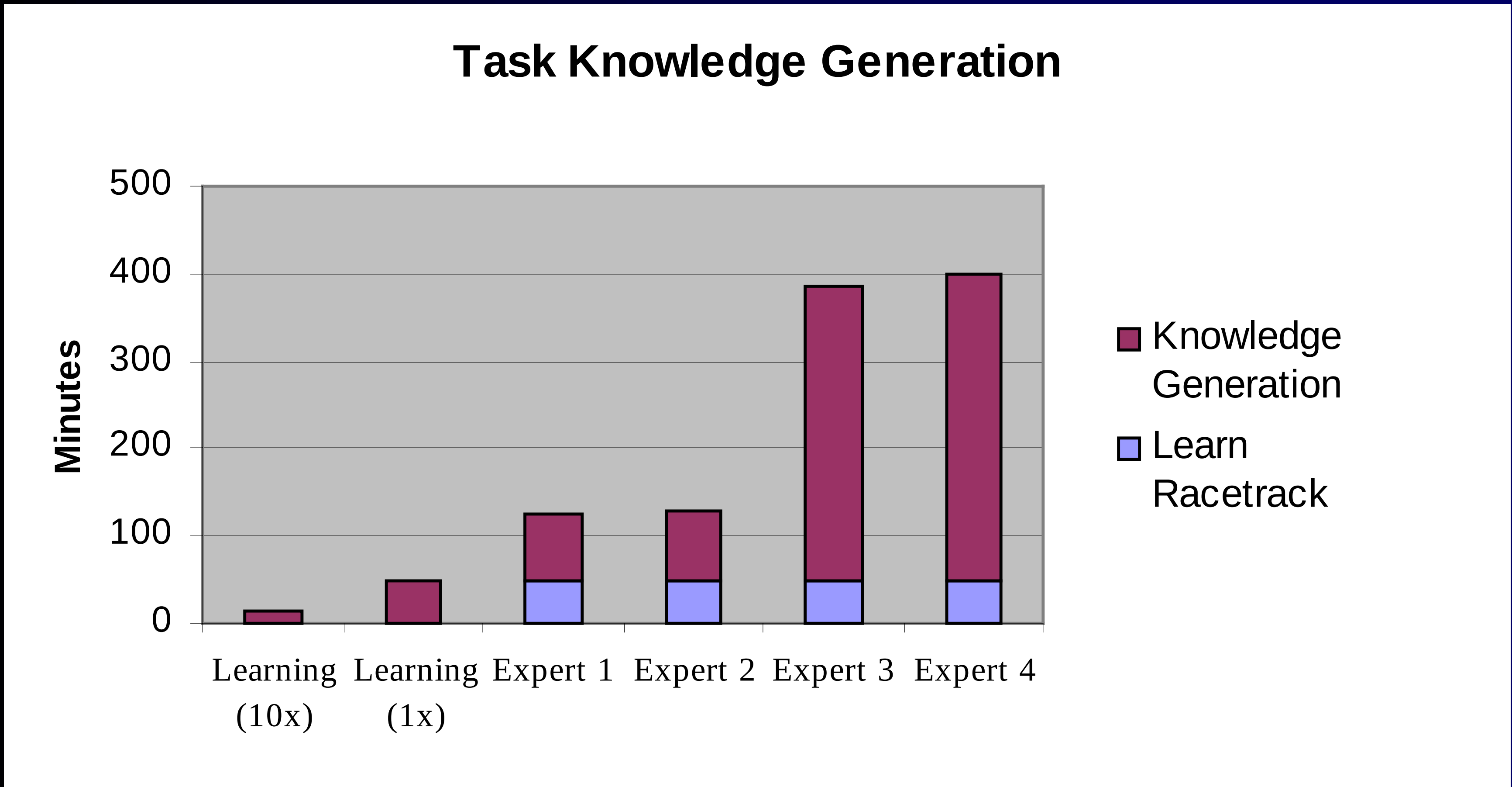


Task Statistics

- Behavior Trace Statistics
  - Racetrack->Racetrack->Intercept->Racetrack
  - 16,000 to 17,000 behavior steps (decision cycles)
  - 30 minutes
  - 23,000 sensor changes
  - 40 actions
  - 31 goal annotations
- Task Performance Knowledge Statistics
  - 85 productions
    - 84% correct, 66% perfect (just like hand coded)
  - 24 operators (4 level hierarchy)
  - learned from 4-8 behavior traces


Productions by Type
Productions
45
40
35
30
25
20
15
10
5
0
Proposal
Action
Goal
Not Working/Untested
Working
Perfect

Behavior Capture in Action



Nuggets and Coal

- Nuggets
  - Complex tasks are being successfully learned
    - Observations of software agents
  - Observation is more efficient than hand-coding
  - Less is required of experts and programmers
- Coal
  - Each behavior must be observed a few times
  - Isn't robust enough to handle human experts well
    - Timing of actions and annotations
    - Depends on reliable responses to goals
  - Don't know if we can learn everything required

Future Challenges

- Improving KnoMic to correctly learn all of intercept
  - Improve interface and learning algorithm
- Improving robustness for human interactions
  - Hard to distinguish errors in behavior vs. delays by expert
- Automatically learning hierarchy
  - Can we eliminate need for human to annotate behavior?