Learning By Instruction Using a Constrained Natural Language Interface

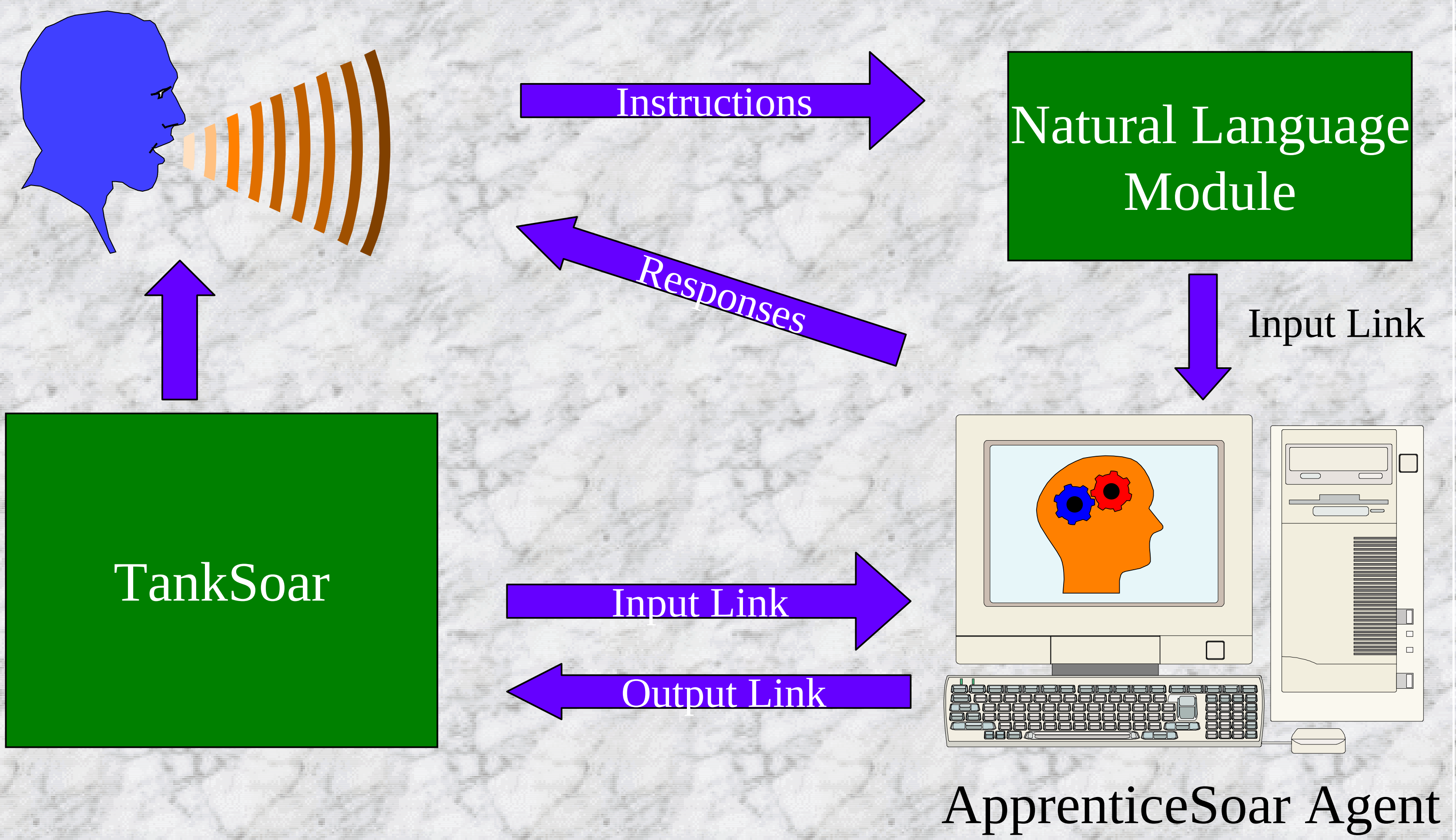
Mazin Assanie and John Laird
Artificial Intelligence Laboratory
The University of Michigan
1101 Beal Ave.
Ann Arbor, Michigan 48109-2122
{mazina,laird}@umich.edu

What is this project about?

- This research project presents an approach to agent learning using interactive natural-language tutorial instruction.

"Always two there are: a master and an apprentice." - Yoda

ApprenticeSoar System Overview



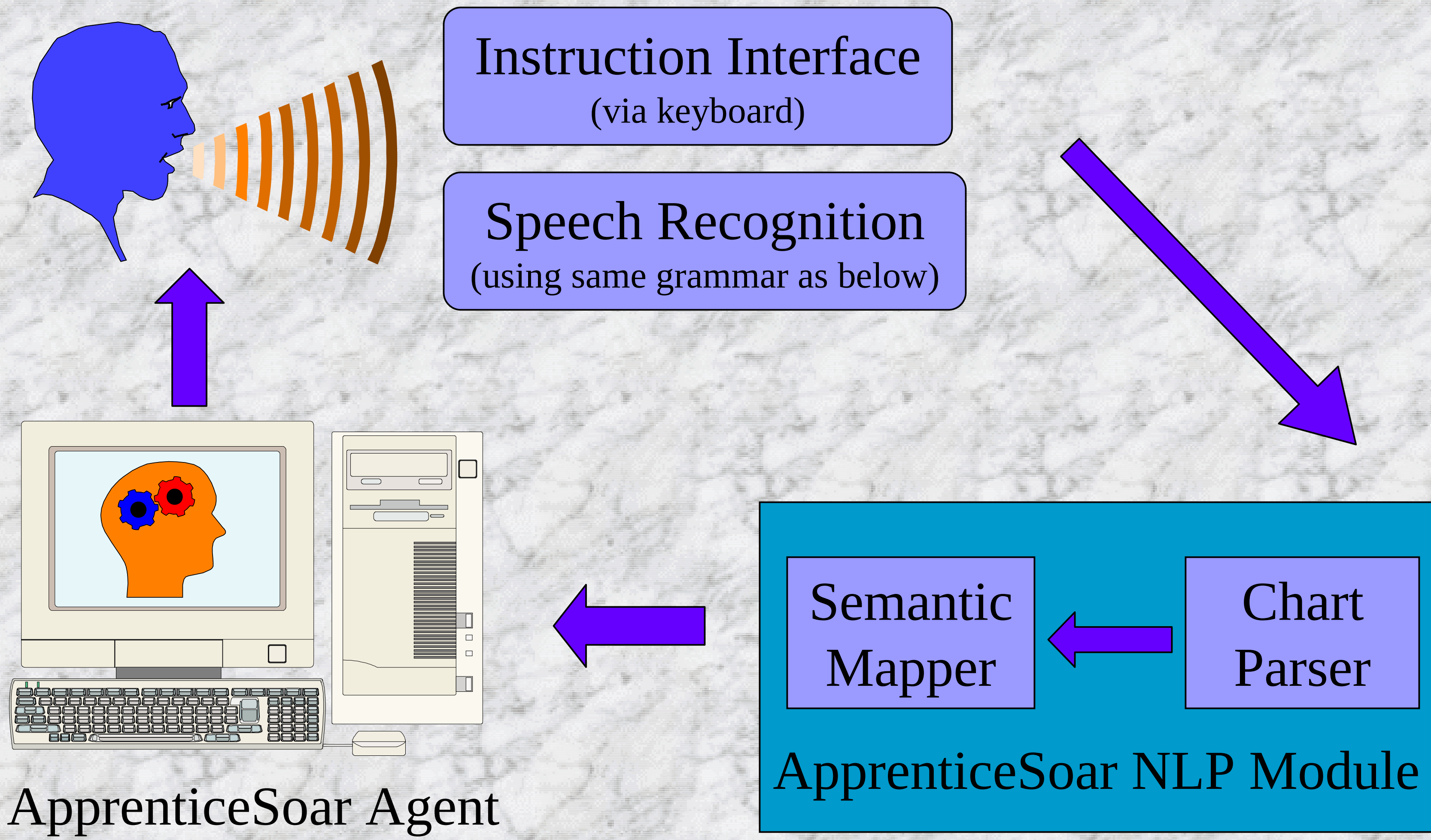
Instructo-Soar

- Scott Huffman and John Laird [1990]
- Contributions towards our work
  - Abstract problem statement and analysis
    - Delineating problem components
    - Classifying missing knowledge types
    - General approach to learning procedural knowledge

Comparing ApprenticeSoar with Instructo-Soar

- Why aren't we using Instructo-Soar's code
  - Conceptually separate learning by instruction
  - All facets deeply intertwined with NL-Soar
    - Result: Recall and learning very different
- Different NLP approach
- Improving breadth of interaction*
- Learning state features
- Learning about dynamic environments
- Achieve same high-level functionality
  - * Currently tackling
  - ** Future milestone

NLP System Overview



Constrained NLP

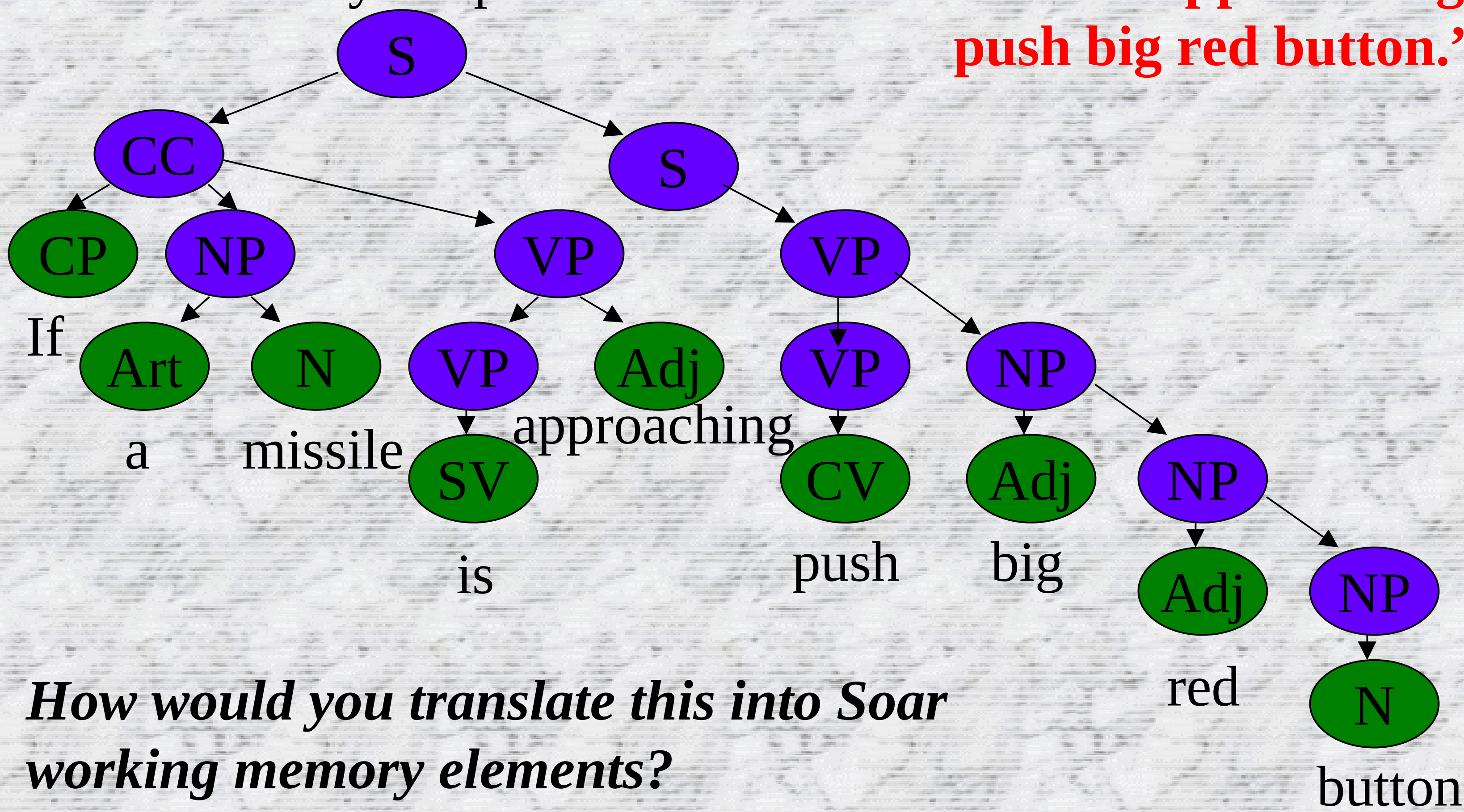
- Use *constraints of dialogue task* to determine range of acceptable utterences.
  - Focus on utility
  - Avoid ambiguity
- User’s burden
- System’s burden
- Is this realistic?
  - Infocom parser

Approach Taken w/ NLP Module

- Goals:
  - Conceptually separate from learning by instruction
  - Fast
  - Easily expandable, modifiable
  - Can be used in your system or other domains
- Why we didn’t use NL-Soar
- Augmented context-free semantic grammar
  - Can use features to increase accuracy
  - Semantic annotation
- Lexicon
  - Contains features and semantic annotations

Translating Parse Trees to Soar WMEs

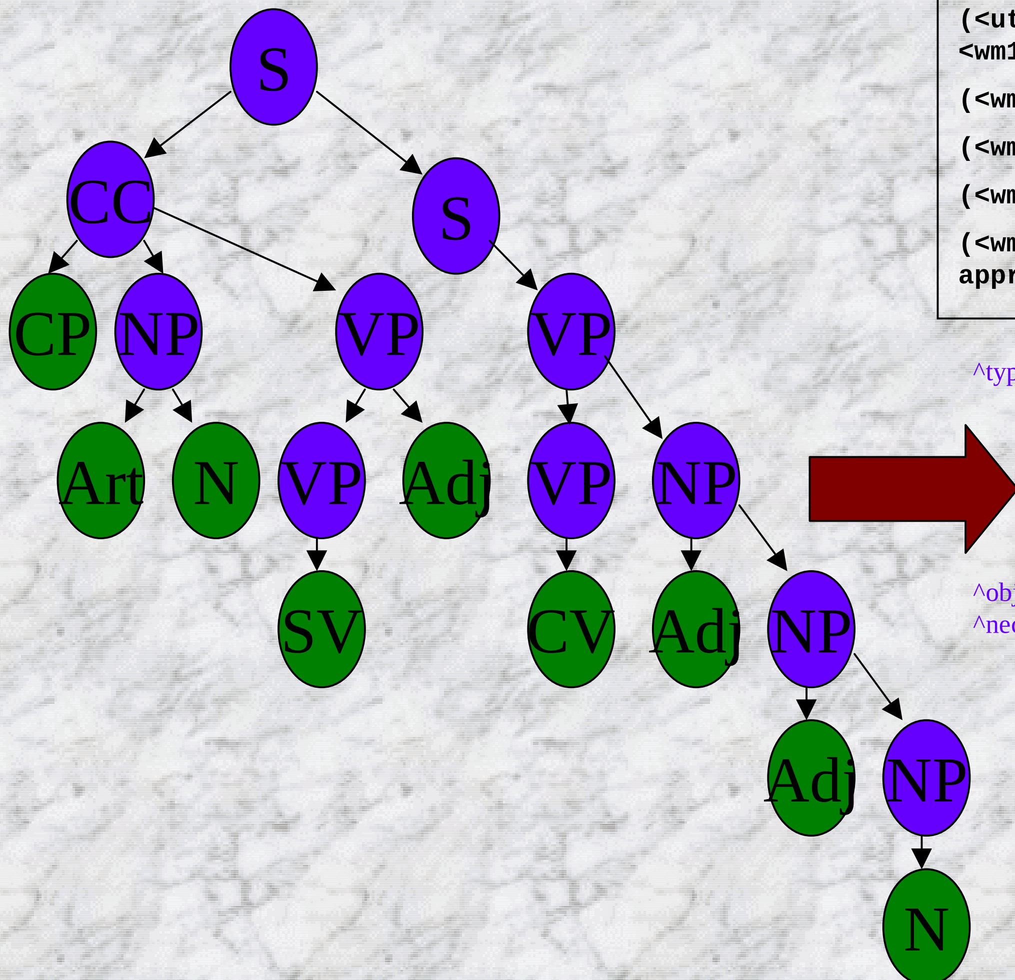
Here's a fairly simple utterance: **"If a missile is approaching, push big red button."**

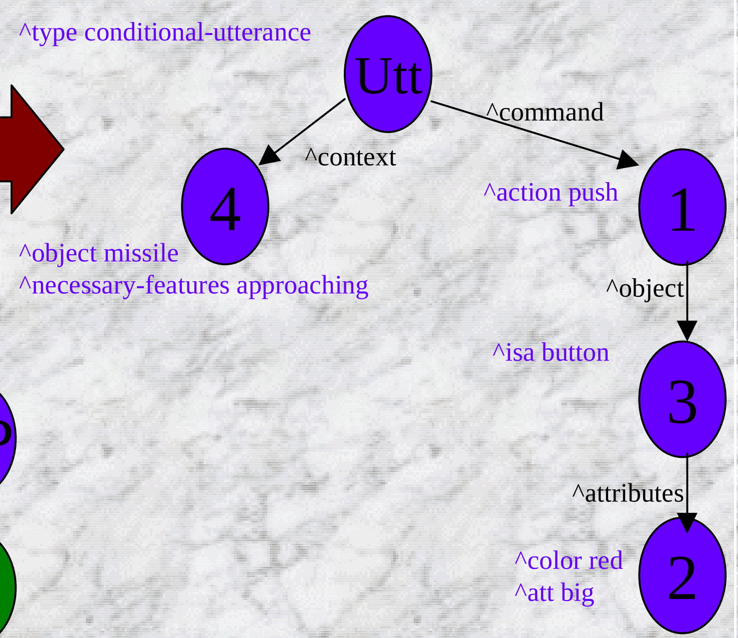


How would you translate this into Soar working memory elements?

Translating Parse Trees to Soar WMEs (cont.)

```
(<utt> ^type conditional-utterance ^command
<wm1> ^context <wm4>)

(<wm1> ^action push ^object <wm3>)

(<wm3> ^isa button ^attributes <wm2>)

(<wm2> ^color red ^att big)

(<wm4> ^object missile ^necessary-features
approaching)
```





Translating Parse Trees to Soar WMEs (cont.)

- Our solution:
  - Semantic annotations on each rewrite rule:
    - Creation of Soar objects
    - Creation of attributes:
      - Literal values
      - Inherited values passed up from children
      - Values from workspace (future work)
  - Bottom-up semantic translation
    - Inheritance rules
    - Creation and linking together of Soar WME's as necessary

A Sample Grammar (sans features)

```
◆ NP = noun [object-class <s1>].
◆ NP = article NP [specific <s1> * <s2>].
◆ NP = adj NP [attribute <s1> * <s2>].

◆ CommandPhrase = CommandVerb [action <s1>].
◆ CommandPhrase = CommandVerb adv [action <s1> attribute <s2>].
◆ CommandPhrase = CommandVerb NP [action <s1> object <<s2>>].

◆ ConditionalClause = ConditionPrep FeatureStatement [* <s2>].


◆ Sentence = CommandPhrase [type imperative-utterance command <<s1>>].
◆ Sentence = ConditionalClause Sentence [conditional-context <<s1>> * <s2>].

◆ * Sentence.
```

context-free rewrite rule

semantic annotation

Soar Component: What do we want the ApprenticeSoar Agent to learn?

- State Features
- Plans
  - Learn new procedural hierarchies
  - Generalizing and extending
  - Effects of operators
- Operator Preference Knowledge
  - Preference
  - Aversion

General Approach to How We Learn From Instruction

- 1. Tutorial dialogue
- 2. Initially stored in short-term memory (WME’s).
- 3. Transferred to long-term memory (chunks)
  - Practice
  - Internal projection

Simple Example of Learning of a State Feature

Tutor: If a missile is approaching, turn-on shields.

Apprentice: I don't know what it means for a missile to have the feature *approaching.* How can I detect it from my percepts?

Tutor: Some incoming sensor is active.

Apprentice : Ok, what other constraints are there?

Tutor: Description-completed.

Apprentice : OK, I have learned a rule to detect when a missile is approaching.

Apprentice : I have learned a general rule that whenever a missile is approaching, I should turn on my shields.

- missile approaches

Apprentice : I have learned a rule for this situation, I am turning on my shields.

Learning State Features

- Learning new mappings
  - Naming state features
  - Renaming plans
- State feature detection dialogue
  - Juxtaposition of sensory constraints
  - Expressibility
  - Declarative to long-term memory

Expressibility of Constraint Specification

- Can apply to either percepts or detected features
- Constraint on a feature
  - Equality
  - Numerical
  - Existence or absence of a feature
- Constraint on feature tree or multiple features
  - Conjunctions
  - All, every, some/any, no
  - Limited disjunction of constraints

Example of Extending A Plan*

- Agent is doing some battle technique. It has lined up another tank and his site and is about to fire.

Apprentice: I cannot proceed with performing battle with blue tank because a there does not exist any missile and that is a precondition of firing.

Tutor: If you do not have any missile, go to weapons recharger.

Apprentice: I know how to do that. Thanks for the advice.

- Agent stores this information declaratively.

- Agent executes the plan to go to the weapons recharger.

- Noticing that it has satisfied the precondition, it learns a new rule to satisfy this precondition in that situation.

Apprentice : I see that going to the weapons recharger has caused a missile to now exist. Returning to performing battle with blue tank. I'll remember your advice if that should happen again.

Achieving Preconditions*

- Impasse in plan execution due to unsatisfied precondition
- Instructor provides missing knowledge
- Agent learns a situated, opportunistic operator that can fire during future impasses to achieve preconditions.

Example of Learning of a New Plan*

Tutor: Go to an energy recharger.

Apprentice: I know how

Tutor: If an energy recharger is ahead, move forward.

Apprentice : Ok, I detect an energy recharger ahead, so I will move forward.

-> Agent moves forward

Apprentice : Continue moving forward until energy recharger not ahead.

-> Agent moves forward twice more. It is on recharger, so no longer ahead.

Apprentice : Ok, I do not detect an recharger ahead. What now?

Tutor: Task-completed.

Apprentice : Thank you. I'll remember how to do that in the future.

Plan Representation

- Plans can contain
  - Nested plans
  - Conditional execution, i.e. branching, loops*
  - Multiple branches**
- Operator templates
  - Goal-oriented
    - Very important because of EBL nature of learning
  - Reduces complexity of planning
  - Must have templates for all primitive operators

Generalizing a New Plan*

- First learn rote lock-step plan
- EBL approach to explain why each step leads to the appropriate goal condition.
  - Must determine goal conditions
    - Hard credit assignment problem.
    - Domain knowledge, instructor assistance/verification
  - 2 Ways to do
    - Internal projection
    - Practice

Learning Operator Preference Knowledge**

- Preferring one operator or plan over another
- Always avoiding certain actions
- Can be useful for plans with multiple non-preferred branches
  - Can learn to choose certain branches depending on the context in which the agent is in
  - Agent would know when to ask for more knowledge

Nuggets

- Sum is greater than its parts
- Must be very careful with chunking
  - Do not want long-term knowledge is dependent upon having the instruction in short-term memory.
- Using a context-free grammar buys you good speech recognition for almost free.

Coal

- Avoid general natural language comprehension.
- No compelling reason to build an agent from scratch using only natural language input.
- Sensitivity to:
  - structure of semantic translation
  - declarative representations
- Implementing the functionality of a large project in a different way requires very careful planning of representations.

Future Work

- Finish items marked with asterisks
- Expand interaction and transfer components
  - Increasing types of interactions/instructions
  - Describing more difficult state features
    - Features over a set of percepts
    - Features resulting from internal reasoning
  - Apprentice-initiated interaction
    - Missing knowledge
    - Verification
    - Explanation
- Using means-end analysis to improve learning