Learning General and Efficient Representations of Novel Games through Interactive Instruction

James Kirk, John Laird

Accepted Paper ACS 2016

Soar Workshop 2016

Extensions to Game Learning

1) Expand the diversity and complexity of tasks to seventeen games

2) Increase the transfer of knowledge between tasks through **hierarchical compositions of concepts**

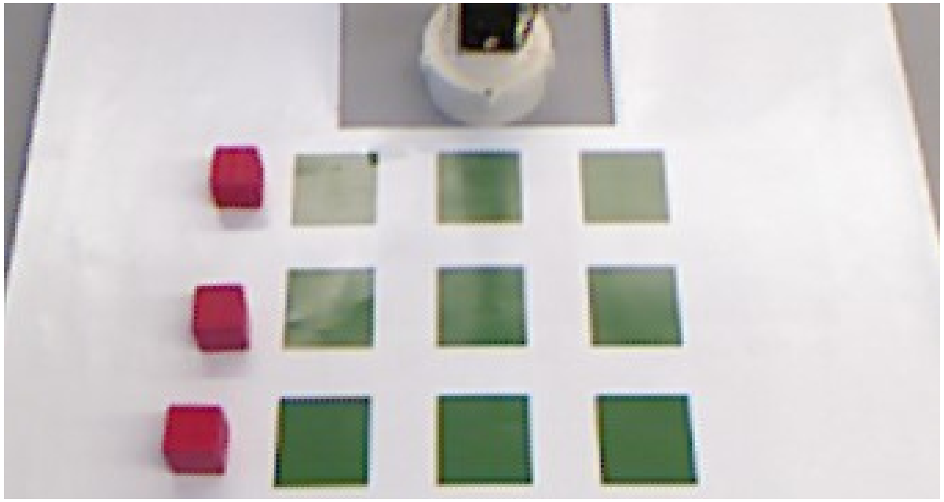
3) Decrease the processing time required to process instructions, learn new tasks, and execute those tasks through **chunking**

4) Allow Rosie to learn many tasks in succession without suffering substantial slowdown as knowledge is acquired

Old vs. New

Previous Rosie:
move 1 on 2
What type of object is parameter 1?
block
Please list all the constraints for this parameter.
it is red
it is not on a location
finished
what type of object is parameter 2?
location
Please list all the constraints for this parameter.
it is not below a location
finished

Extended Version:
You can move a free red block onto a clear location.
I don't know the concept 'free'.
If a block is not on a location then it is free.
Ok, I've learned the concept free.
Does 'clear' mean that it is not below a block?
Yes.



Task Concept Representation

- A task concept: action, goal, failure, or new predicate is defined by
  - A linguistic term ("stacked", "three-in-a-row", "clear")
  - A conjunction of predicate tests (*clear(X) ^ block(X) ^* ….)
  - Usage specific to the type (action -> proposal, goal -> success, failure -> terminal state …)
- Objects
  - Physical objects (o_1,o_2….)
  - Numbers (n_1,n_2….)
  - Strings (s_1, s_2…)
  - Sets of objects (O_1, O_2…)
- Predicates *p(x,….)*
  - Unary features: *red, large, block, location, clear*
  - N-ary relations: *on, behind, between*
- Functions *y= f(x..) represented by predicate p(y,x…)*
  - *number-of, attribute-of, sum*

Concept Learning Process

1. Structure Learning
   a. Natural Language Processing
   b. Declarative Predicate Structure Construction
2. Interpretation Phase
   a. Predicate Matching (Grounding)
   b. Joining (Satisfying)
   c. Application (Usage Matching)
3. Dynamic Compilation through Chunking

We will illustrate this process on the following goal sentence:

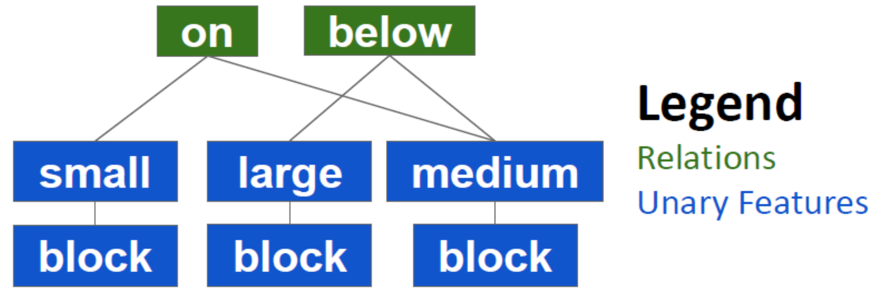
The goal is that a small block is on a medium block and a large block is below the medium block.

Structure Learning

…a small block is on a medium block and a large block is below the medium block

Predicate representation:

$$small(x_1) \land block(x_1) \land \\ medium(x_2) \land block(x_2) \land \\ large(x_3) \land block(x_3) \land \\ below(x_3, x_2) \land on(x_1, x_2)$$



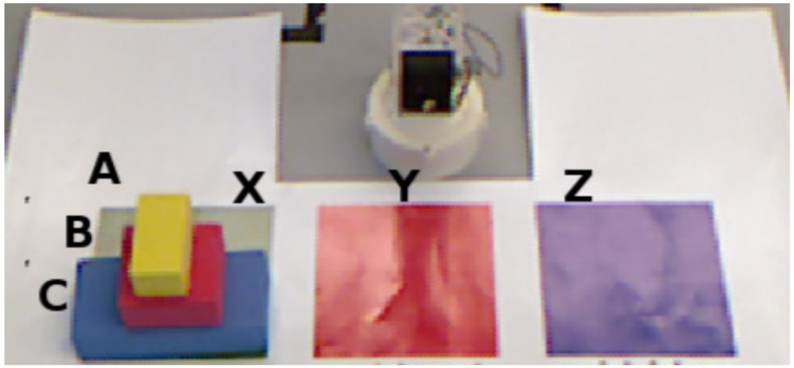
From predicate conjunction a declarative tree structure is built for efficient bottom to top evaluation

For each predicate:

1. Added to structure on top of last reference to tested object
2. Stored as last reference to tested object ($\boldsymbol{x_n}$)
3. Leaf nodes (first references) will be evaluated directly against the world

Iterates through predicates based on arity: unary, binary, n-ary

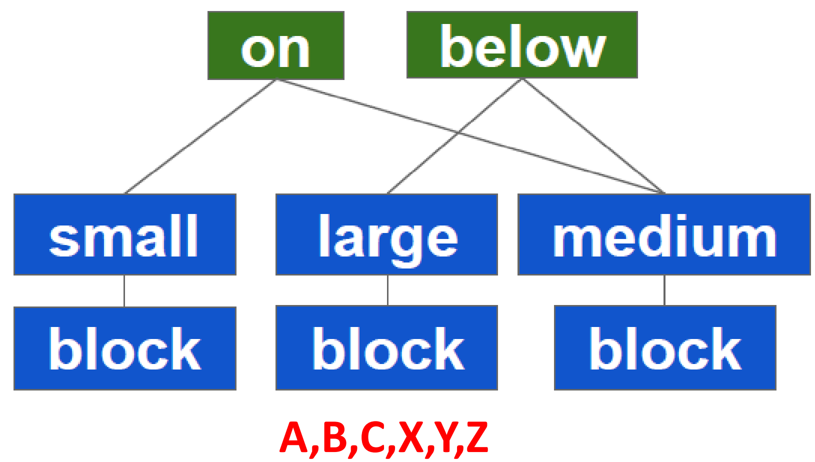
Interpretation



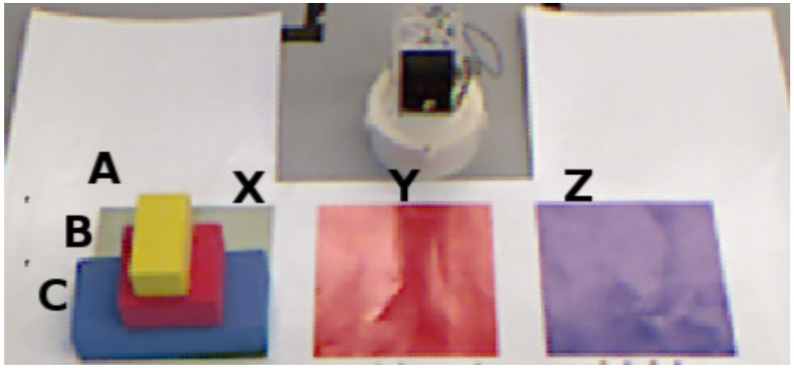
Predicate instances in world state representation:

$small(A), medium(B), large(C)$
$on(A, B), on(B, C), on(C, X)$
$below(B, A), below(C, B), below(X, C)$
$location : \{X, Y, Z\}, block : \{A, B, C\}$

- Predicate Matching
  - Retrieve semantics of predicate based on linguistic term ("number of"-> *count* operator, "red" -> primitive color, "on" -> spatial preposition)
  - Evaluate predicates within context of world state and children in tree

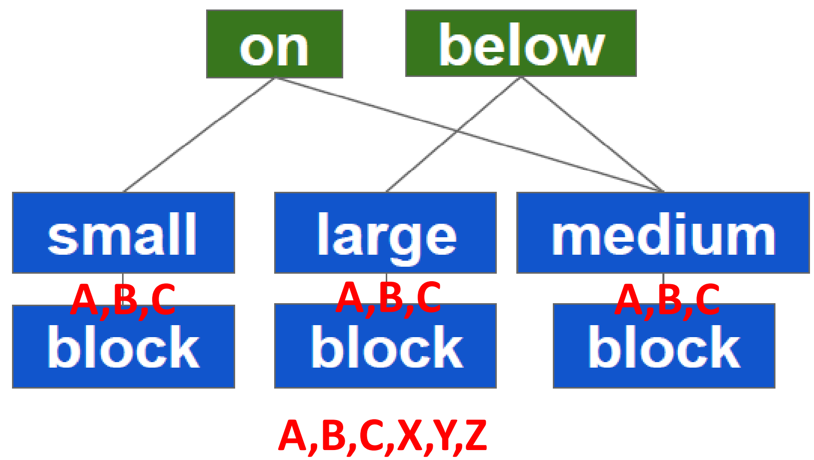


Interpretation

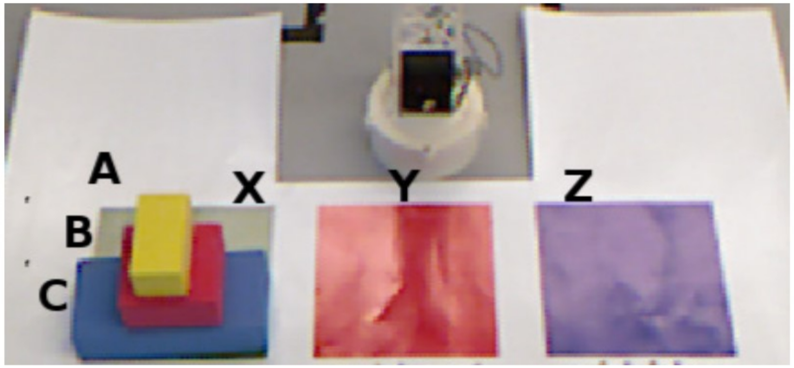


Predicate instances in world state representation:
$small(A), medium(B), large(C)$
$on(A,B), on(B,C), on(C,X)$
$below(B,A), below(C,B), below(X,C)$
$location: \{X,Y,Z\}, block: \{A,B,C\}$

- Predicate Matching
  - Retrieve semantics of predicate based on linguistic term (“number of”-> *count* operator, “red” -> primitive color, “on” -> spatial preposition)
  - Evaluate predicates within context of world state and children in tree
  - Evaluate ***block*** on all world objects (A-Z) **: A, B, C**

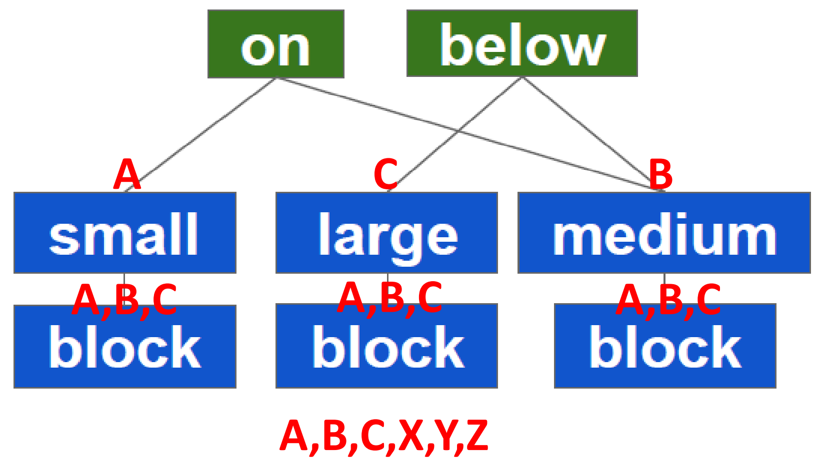


Interpretation

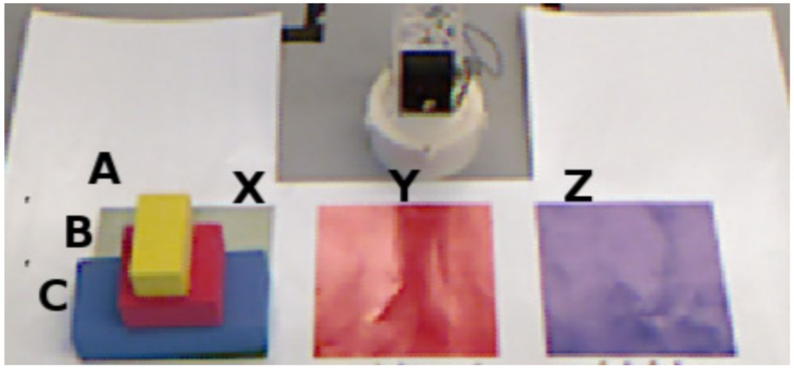


Predicate instances in world state representation:
$small(A), medium(B), large(C)$
$on(A,B), on(B,C), on(C,X)$
$below(B,A), below(C,B), below(X,C)$
$location : \{X, Y, Z\}, block : \{A, B, C\}$

- Predicate Matching
  - Retrieve semantics of predicate based on linguistic term ("number of"-> *count* operator, "red" -> primitive color, "on" -> spatial preposition)
  - Evaluate predicates within context of world state and children in tree
  - Evaluate ***block*** on all world objects (A-Z) **: A, B, C**
  - Evaluate ***small*** on A, B, C **: A**

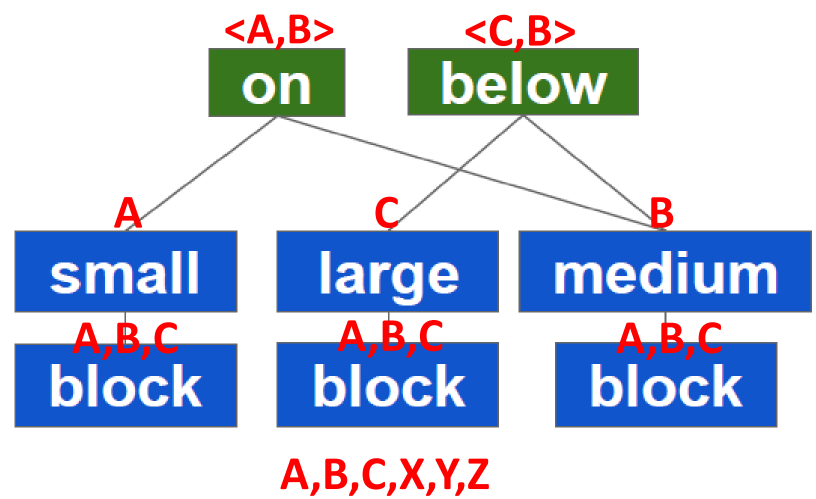


Interpretation

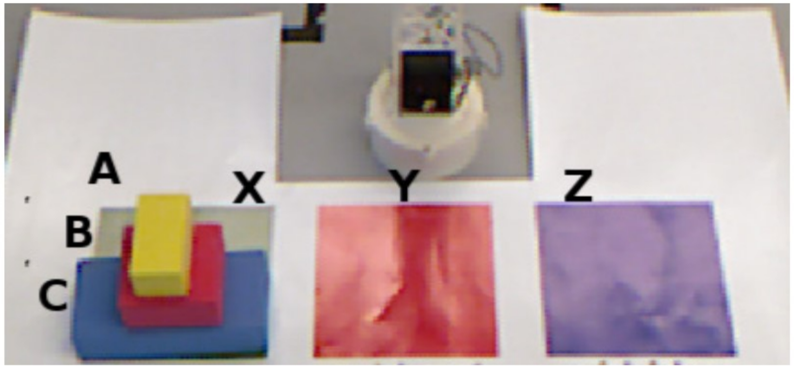


Predicate instances in world state representation:
$small(A), medium(B), large(C)$
$on(A, B), on(B, C), on(C, X)$
$below(B, A), below(C, B), below(X, C)$
$location : \{X, Y, Z\}, block : \{A, B, C\}$

- Predicate Matching
  - Retrieve semantics of predicate based on linguistic term ("number of"-> *count* operator, "red" -> primitive color, "on" -> spatial preposition)
  - Evaluate predicates within context of world state and children in tree
  - Evaluate ***block*** on all world objects (A-Z) **: A, B, C**
  - Evaluate ***small*** on A, B, C **: A**
  - Evaluate ***on*** on (A,B) : <**A,B**>

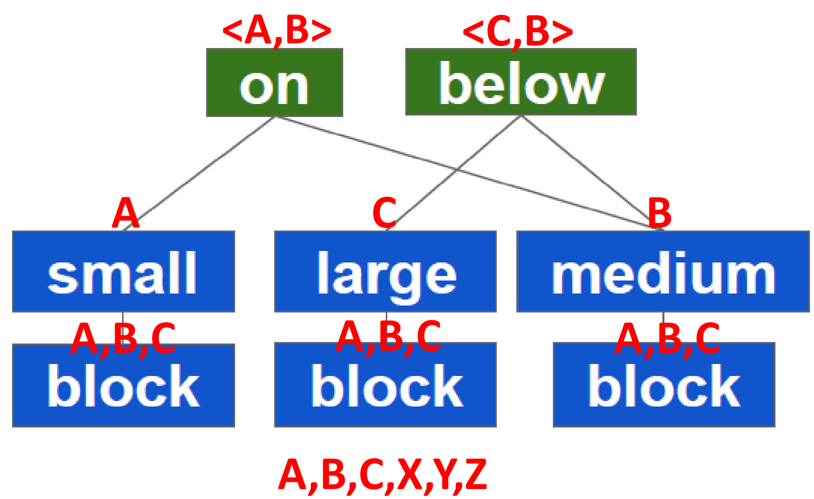


Interpretation



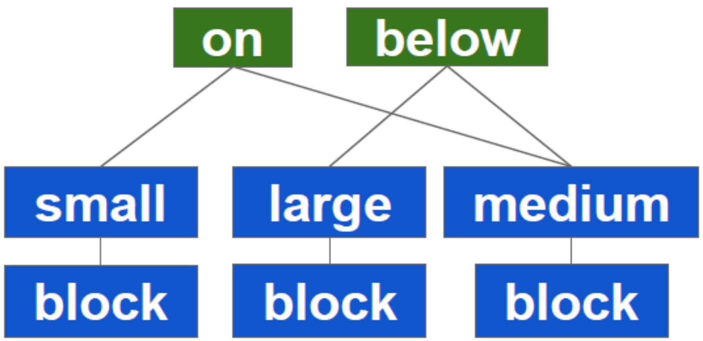
Predicate instances in world state representation:
$small(A), medium(B), large(C)$
$on(A, B), on(B, C), on(C, X)$
$below(B, A), below(C, B), below(X, C)$
$location : \{X, Y, Z\}, block : \{A, B, C\}$

- Predicate Matching
  - Retrieve semantics of predicate based on linguistic term (“number of”-> *count* operator, “red” -> primitive color, “on” -> spatial preposition)
  - Evaluate predicates within context of world state and children in tree
  - Evaluate ***block*** on all world objects (A-Z) **: A, B, C**
  - Evaluate ***small*** on A, B, C **: A**
  - Evaluate ***on*** on (A,B) : <**A,B**>
- Joining, Satisfying
  - Evaluate intersection of results from predicate matching
  - Result is the objects and values that satisfy all constraints
- Application
  - Goal: detection of goal/winning
  - Action: proposal of available actions
  - Failure: detection of terminal state/losing
  - New predicates: successful predicate match



Dynamic Compilation: Chunking

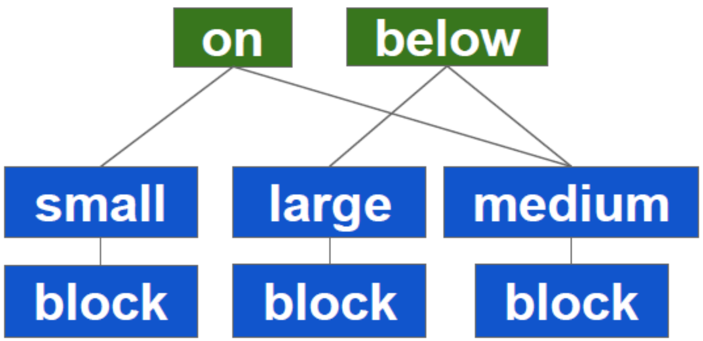
- Predicate Matching




```
sp {chunk*justification-641*t1279-1
     :chunk
     (state <s1> ^gtype <c2> ^<c2> <a1>)
     (<a1> ^condition <c3>)
     (<c3> ^name <block1> ^attribute <category> ^rtype single
           ^type attribute ^args <a2> ^parameter <p1>
           ^result.set <p2>)
     (<p1> ^num { < 2 <c7> })
     (<a2> ^1 <c8>)
     (<c8> ^result.set <r3>)
     (<r3> ^instance <i1>)
     (<i1> ^1 <n1>)
     (<n1> ^predicates <p3>)
     (<p3> ^<category> <block1>)
     -->
     (<p2> ^instance <i2> +)
     (<i2> ^1 <n1> +)
}
```

Dynamic Compilation: Chunking

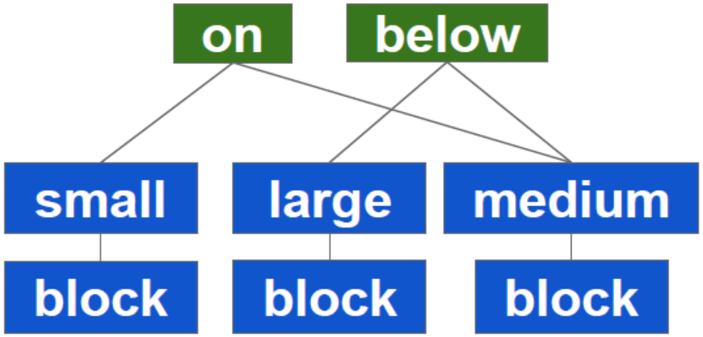
- Predicate Matching




```
sp {chunk*justification-641*t1279-1
    :chunk
    (state <s1> ^gtype <c2> ^<c2> <a1>)
    (<a1> ^condition <c3>)
    (<c3> ^name <small1> ^attribute <size> ^rtype single
          ^type attribute ^args <a2> ^parameter <p1>
          ^result.set <p2>)
    (<p1> ^num { < 2 <c7> })
    (<a2> ^1 <c8>)
    (<c8> ^result.set <r3>)
    (<r3> ^instance <i1>)
    (<i1> ^1 <n1>)
    (<n1> ^predicates <p3>)
    (<p3> ^<size> <small1>)
    -->
    (<p2> ^instance <i2> +)
    (<i2> ^1 <n1> +)
}
```

Dynamic Compilation: Chunking

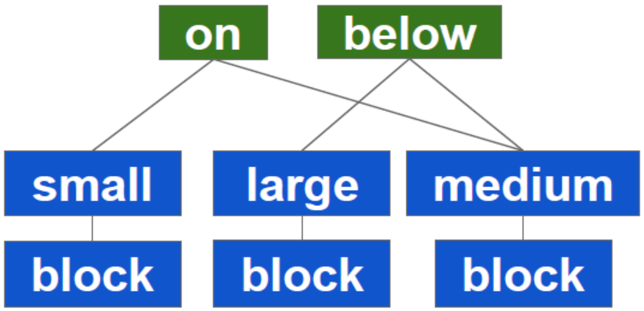
- Predicate Matching




```
sp {chunk*justification-680*t1284-1
    :chunk
    (state <s1> ^list <l1> ^type <goal> ^<goal> <a1>
          ^world <n1>)
    (<a1> ^condition <n6>)
    (<n6> ^name <on1> ^rtype single ^type spatial-preposition ^args <a2>
          ^negative false ^result.set <p5> ^parameter <p2>)
    (<a2> ^num 2 ^2 <c5> ^1 <c6>)
    (<p2> ^num 2)
    (<c5> -^rtype set ^result.set <p3>)
    (<c6> -^rtype set ^result.set <p4>)
    (<l1> ^game <g1>)
    (<n1> ^predicates <p1>)
    (<p1> ^predicate <n2>)
    (<n2> ^handle <c2> ^instance <n3>)
    (<n3> ^2 <n4> ^1 <n5>)
    (<p4> ^instance <i2>)
    (<i2> ^1 <n5>)
    (<p3> ^instance <i1>)
    (<i1> ^1 <n4>)
    -->
    (<p5> ^instance <i3> +)
    (<i3> ^2 <n4> + ^1 <n5> +)
}
```

Chunking

- Application: Term and structure linking




```
sp {chunk-multi*chunk-game-impasse*apply*complete*snc*t2417-2
    :chunk
    (state <s1> ^retrieve-game blocks-world)
    -->
    (<s1> ^retrieve-handle stacked-up2 +)
}

sp {chunk-multi*chunk-predicate-impasse*apply*complete*goal*snc*t2410-1
    :chunk
    (state <s1> ^retrieve-handle stacked-up2 ^type goal)
    -->
    (<s1> ^goal <p1> +)
    (<p1> ^parameter-set <p3> + ^primary-rtype single + ^nlp-set <p13> +
           ^handle stacked-up2 +)
    (<p3> ^argnum 3 +)
    (<p13> ^conditions <n1> + ^conditions <n2> +)
    (<n1> ^type state-pair + ^name on1 + ^attribute prepositions +
           ^result <r1> + ^parameter <p6> + ^negative false + ^args <a4> +
           ^rtype single +)
    (<a4> ^1 <c1> + ^2 <c4> + ^num 2 +)
    (<c1> ^type attribute + ^name small1 + ^attribute size + ^result <r2> +
           ^parameter <p9> + ^negative false + ^args <a3> + ^rtype single +)
    (<a3> ^1 <c2> + ^num 1 +)
    (<c2> ^type attribute + ^name block1 + ^attribute category + ^result <r3> +
           ^parameter <p11> + ^negative false + ^args <a2> + ^rtype single +)
    (<c4> ^type attribute + ^name medium1 + ^attribute size + ^result <r11> +
           ^parameter <p24> + ^negative false + ^args <a11> + ^rtype single +)
    (<a11> ^1 <c8> + ^num 1 +)
    (<c8> ^type attribute + ^name block1 + ^attribute category +
           ^result <r12> + ^parameter <p26> + ^negative false + ^args <a10> +
           ^rtype single +)
    (<a10> ^1 <c9> + ^num 1 +)
.
.
.
}
```

Action Example

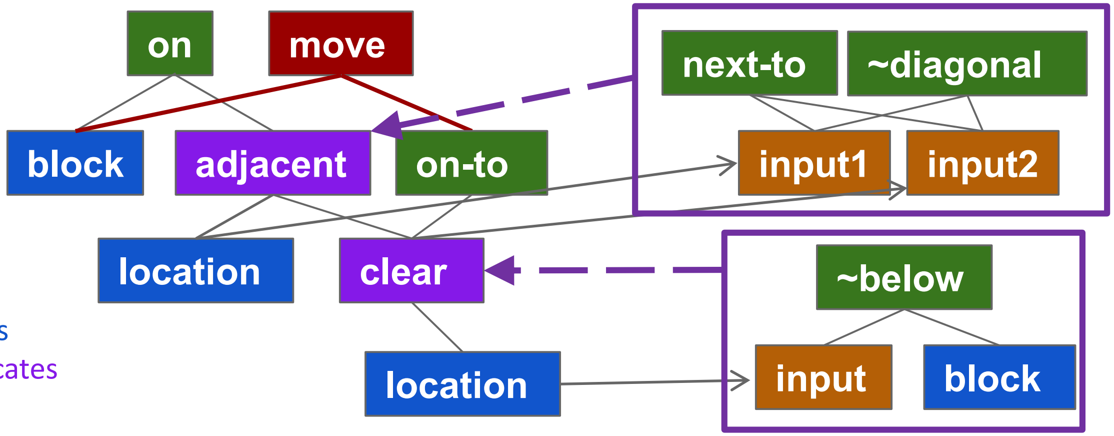
> If a block is on a location that is adjacent to a clear location then you can move the block onto the clear location.

I don't know the concept adjacent.

> If a location is next to a clear location but it is not diagonal with the clear location then it is adjacent to the clear location.

Legend

Verbs
Relations
Inputs
Unary Features
Learned Predicates



Learned Predicates

- From composition of primitives can learn many new typs of knowledge
- Learned concepts can be
  - General across domains(clear)
  - Task/domain specific (matched, yours)
  - Mapping is many-to-many
- Prepositions
  - adjacent
- Labels
  - captured, your, current
- Functions
  - passenger-of, husband-of
- Synonynms, Antonyms, and Homonyms
  - huge, crimson
  - clear and covered
  - matched

If a location is below your block then it is ***captured***.

If a block is red then it is ***your*** block.

If a bank is below the boat then it is the ***current*** bank.

If an object is on a boat then it is a ***passenger of*** the boat.

If the last-name of a woman is the last-name of a man then the man is the ***husband of*** the woman.

If a block is large then it is ***huge***.

If an object is below a block then it is ***covered***.

If the value of a location is the value of the tile that is on the location then the location is ***matched***.

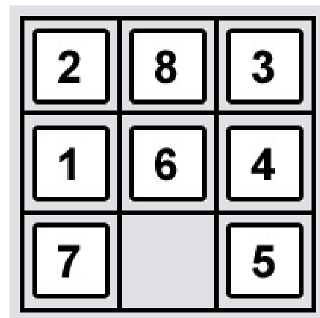
If the color of a location is the color of the block that is on the location then the location is ***matched***.

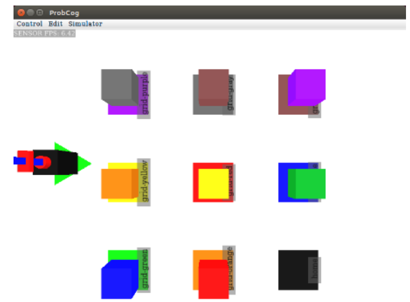
More Sentence Examples

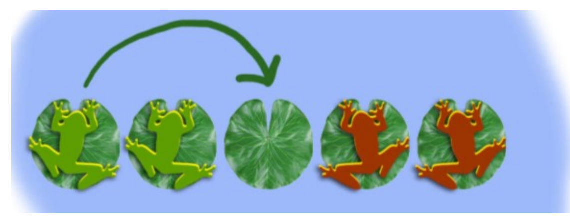
1. You can move a clear block onto a clear location. [Blocks World]
2. The goal is that there are eight matched locations. [Eight Puzzle]
3. If the number of cannibals on a bank is more than the number of missionaries on the bank then you lose. [Missionaries and Cannibals]
4. The goal is that all locations are covered and the number of captured locations is more than the number of occupied locations. [Othello]
5. If the locations between a clear location and a captured location are occupied then you can move a free red block onto the clear location. [Othello]
6. If a woman is on a bank and the husband of the woman is not on the bank and another man is on the bank then you lose. [Jealous Husbands]
7. You can move a passenger of the boat onto the current bank. [Fox Puzzle]
8. The goal is that all the red blocks are on the red locations and all the blue blocks are on the blue locations. [Frogs and Toads]

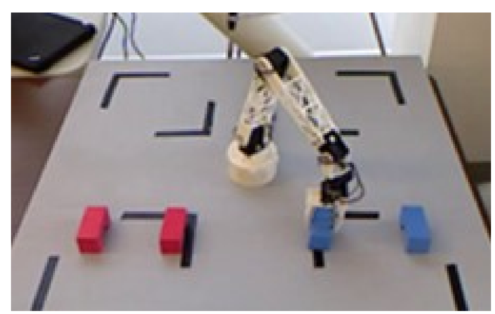
17 Games

- Missionaries and Cannibals
- Jealous Husbands problem
- Frogs and Toads puzzle
- Eight Puzzle
- Five Puzzle
- Tower of Hanoi (3 blocks)
- Tower of Hanoi (4 blocks)
- Fox Puzzle
- Tic-Tac-Toe
- Othello
- Three Men’s Morris
- Picaria
- Nine Holes
- Simplified Risk
- Mahjong Solitaire
- Simple Maze
- Blocks World

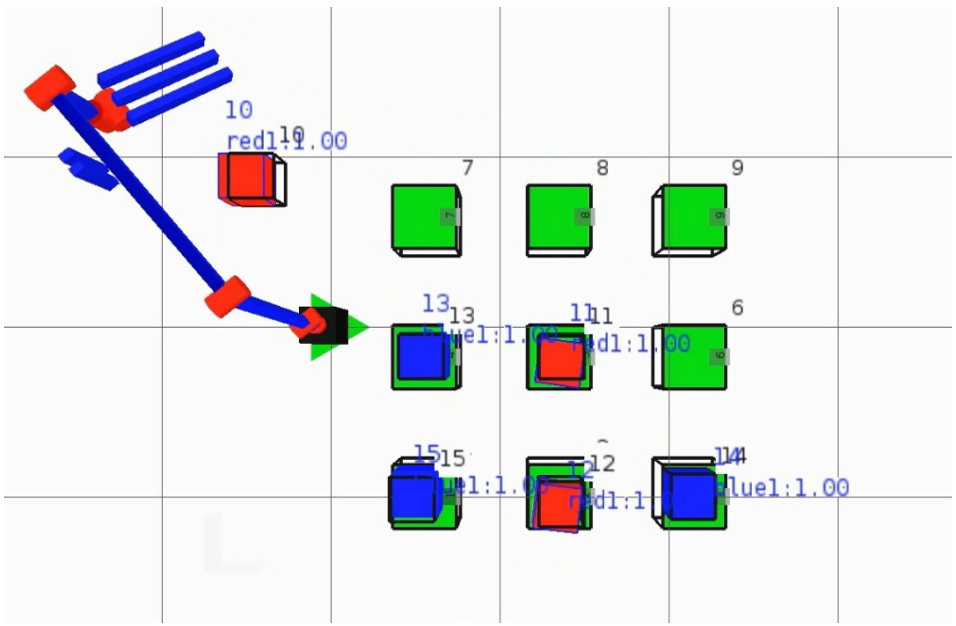


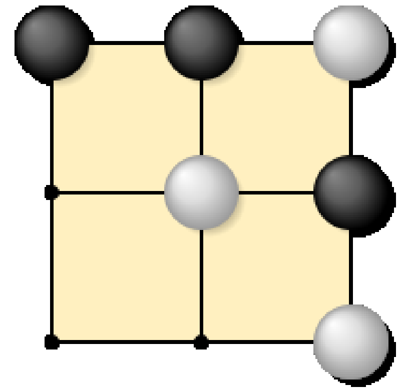


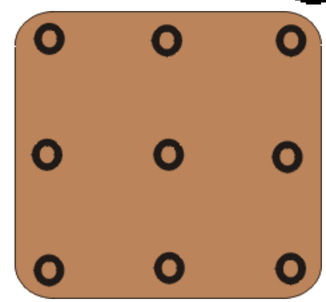


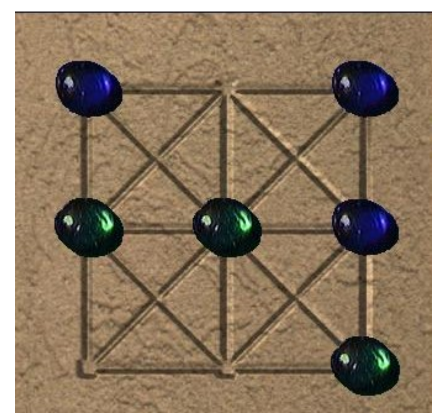


3x3 Board Games

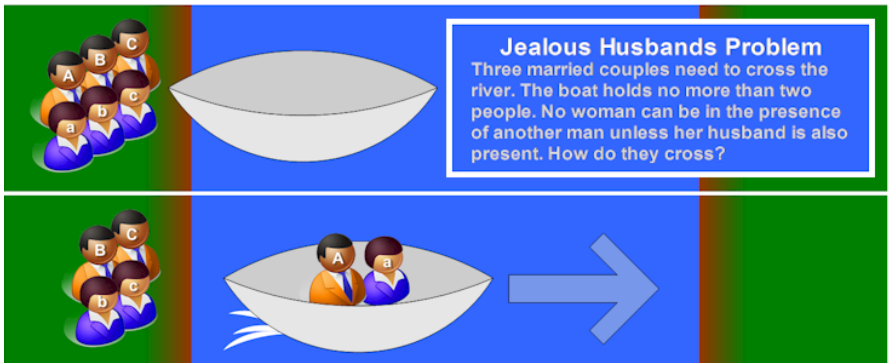


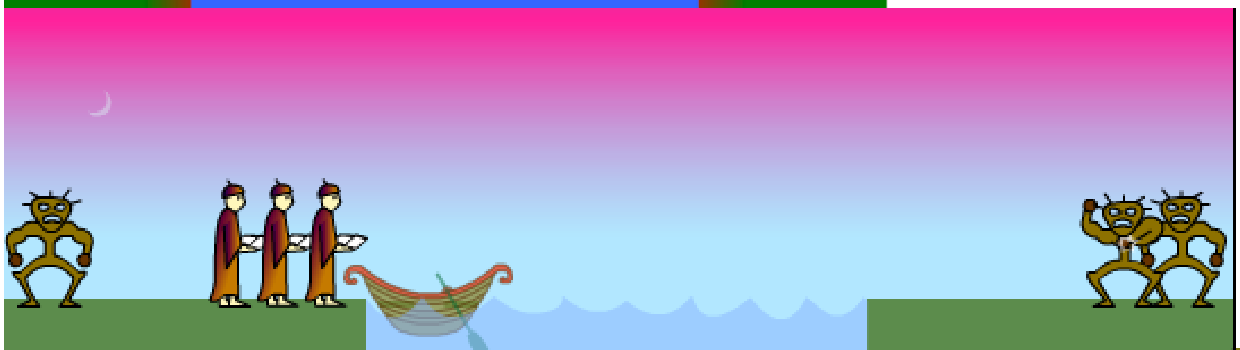






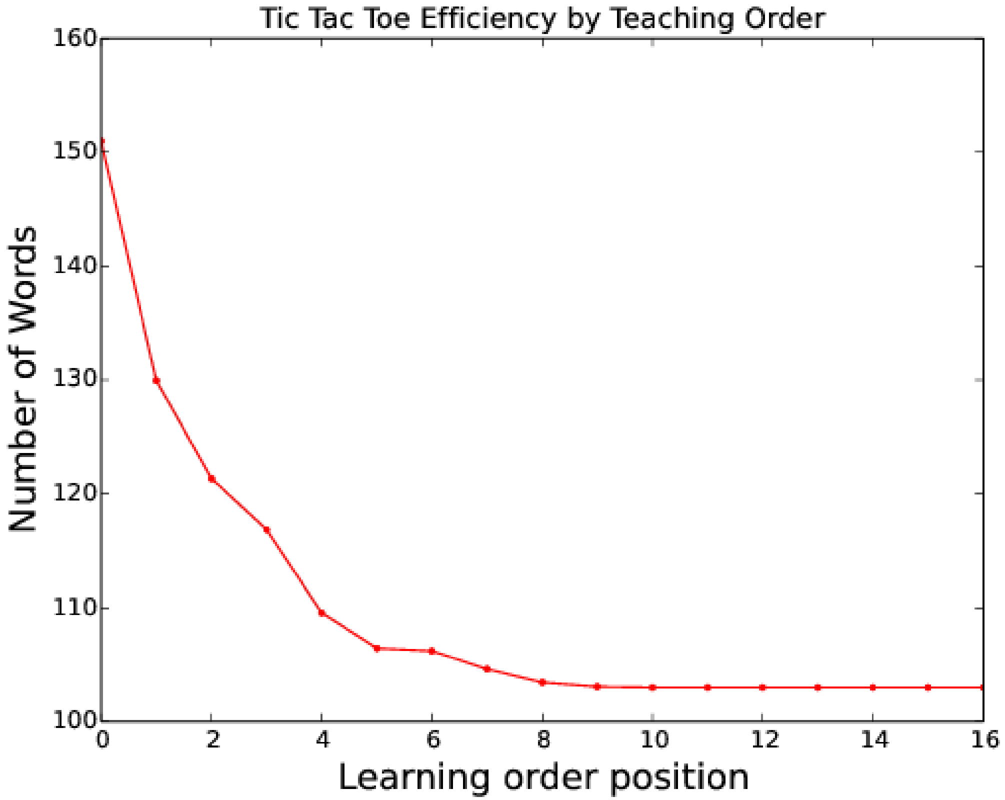
River Crossing Puzzles



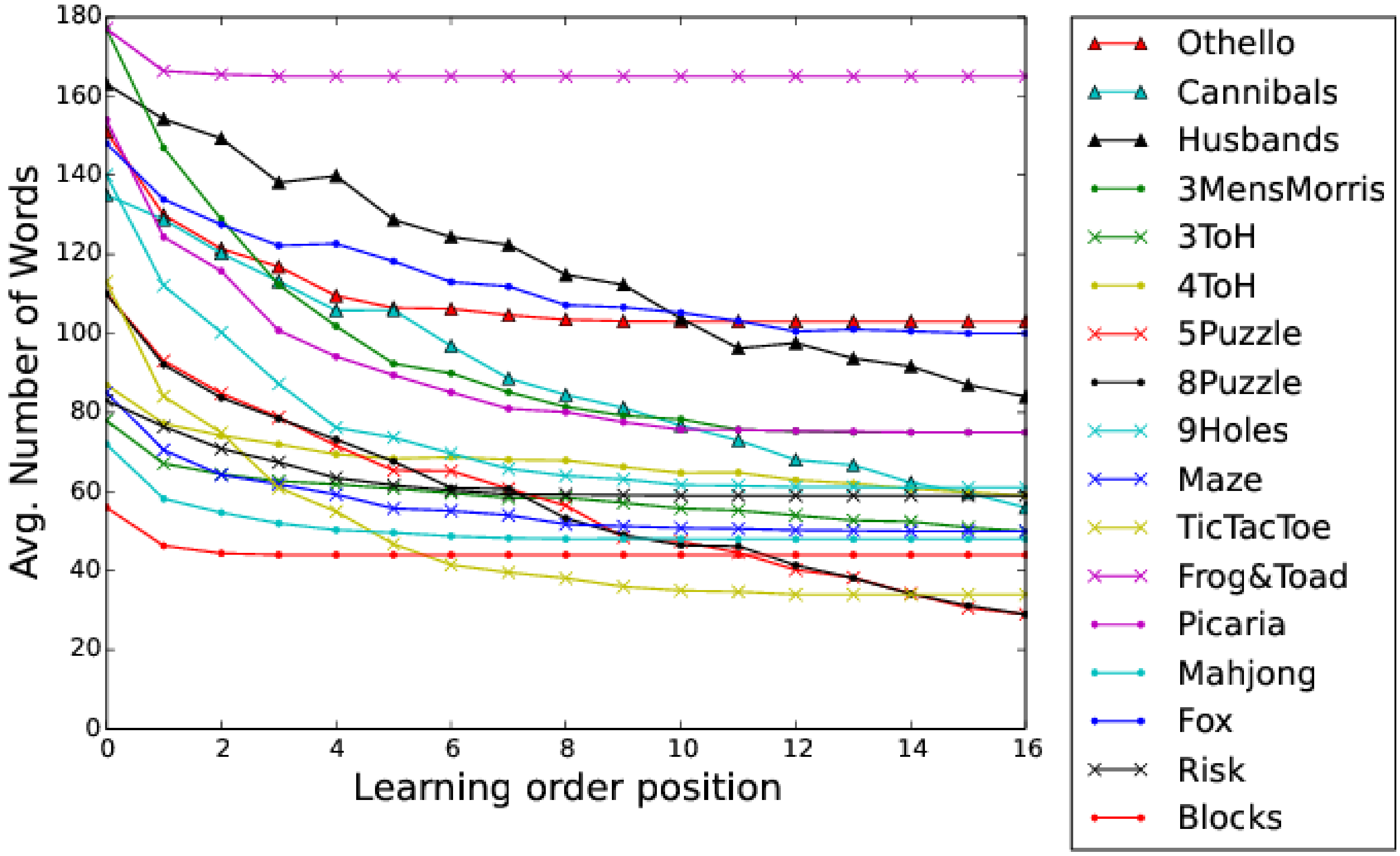


Evaluation

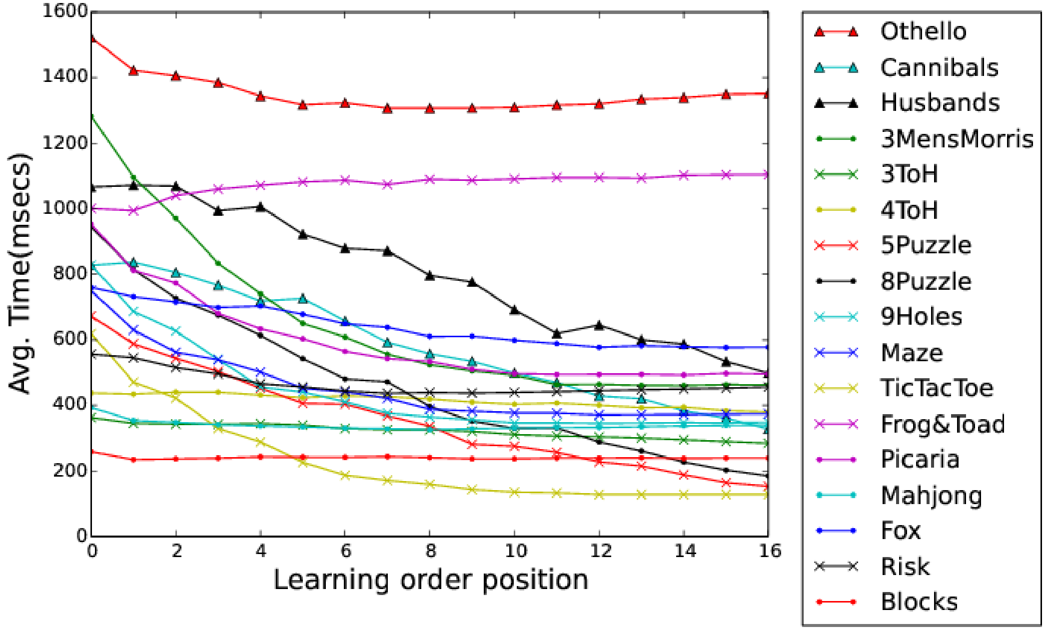
- 3000 randomly generated permutations of 17 games
- Scripted language, simulated symbolic domain
- Analyze efficiency and the affects of order (transfer)
  - Communication time (number of words)
  - Processing time
  - Memory sizes

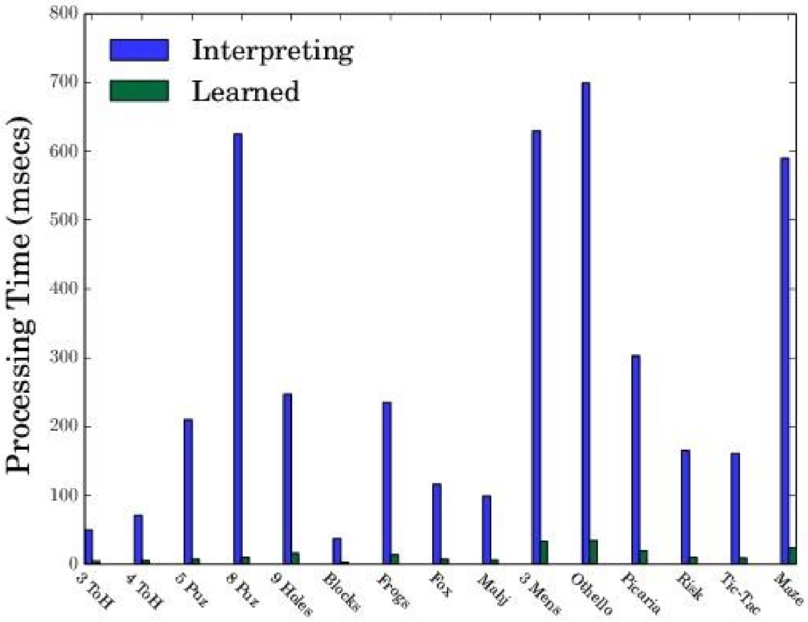


Communication Efficiency

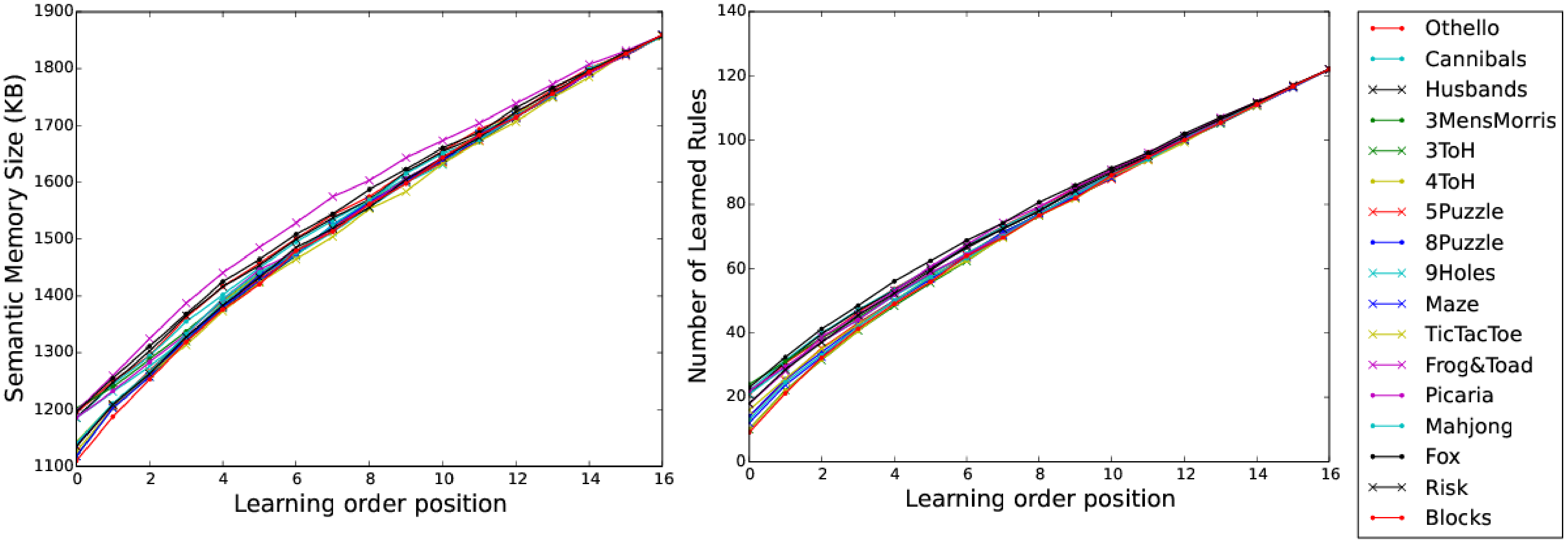


Processing Efficiency

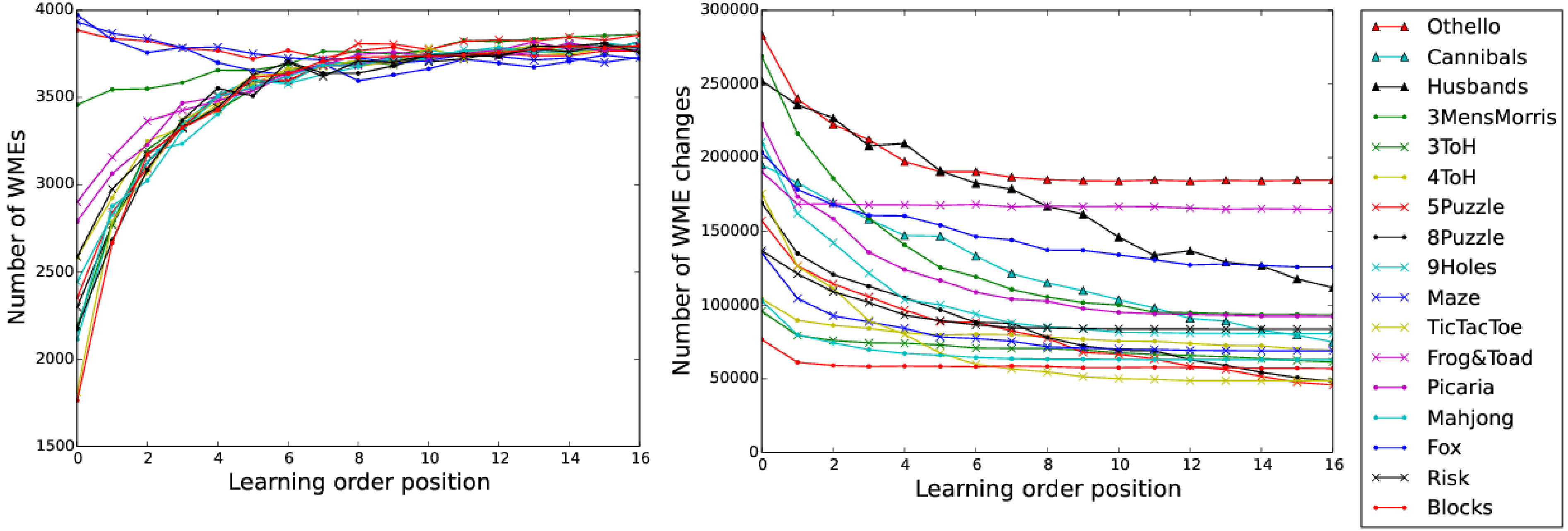




Semantic and Procedural Memory



Working Memory



Nuggets and Coals

Nuggets

- Demonstrates generality over many games and puzzles
- Hierarchical composition of concepts allows the definition of many new concepts and improves efficiency of communication
- Variablized Chunking drastically improves efficiency of processing
- No substantial slowdown over many successive tasks

Coals

- Doesn’t learn policy or heuristic knowledge, only iterative deepening search
- Does not scale well to handle large number of objects
- Not robust to errors in teaching

Questions?