Learning First-Order Definite Theories via Object-Based Queries

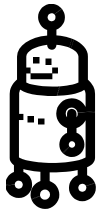
Joseph Selman and Alan Fern
School of EECS, Oregon State University

Motivation

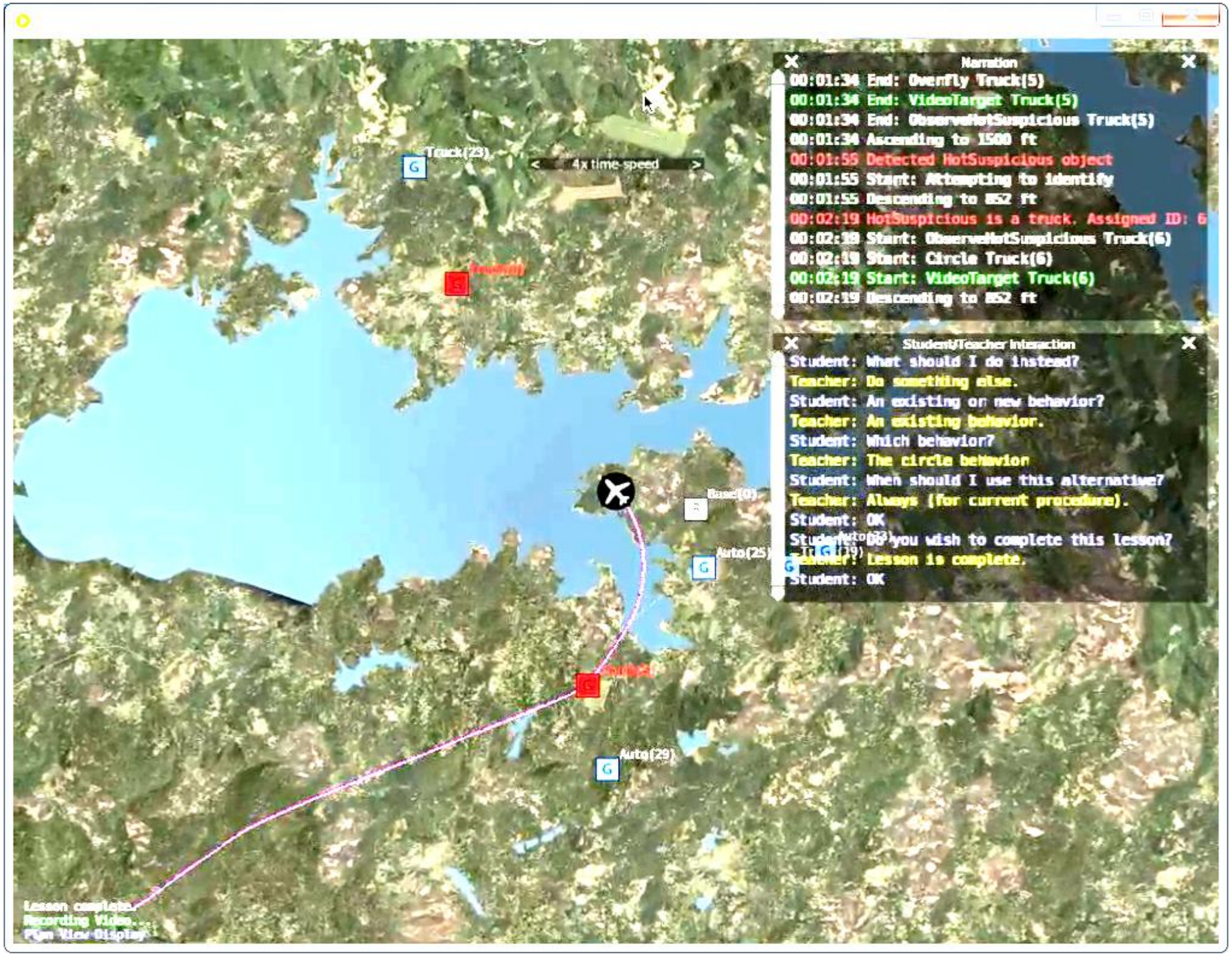
- Enable systems to learn from natural instruction methods
  - From a theoretical perspective

- Assumptions about teachers
  1. They have a good understanding of the target concept
  2. Non-experts in knowledge representation and ML

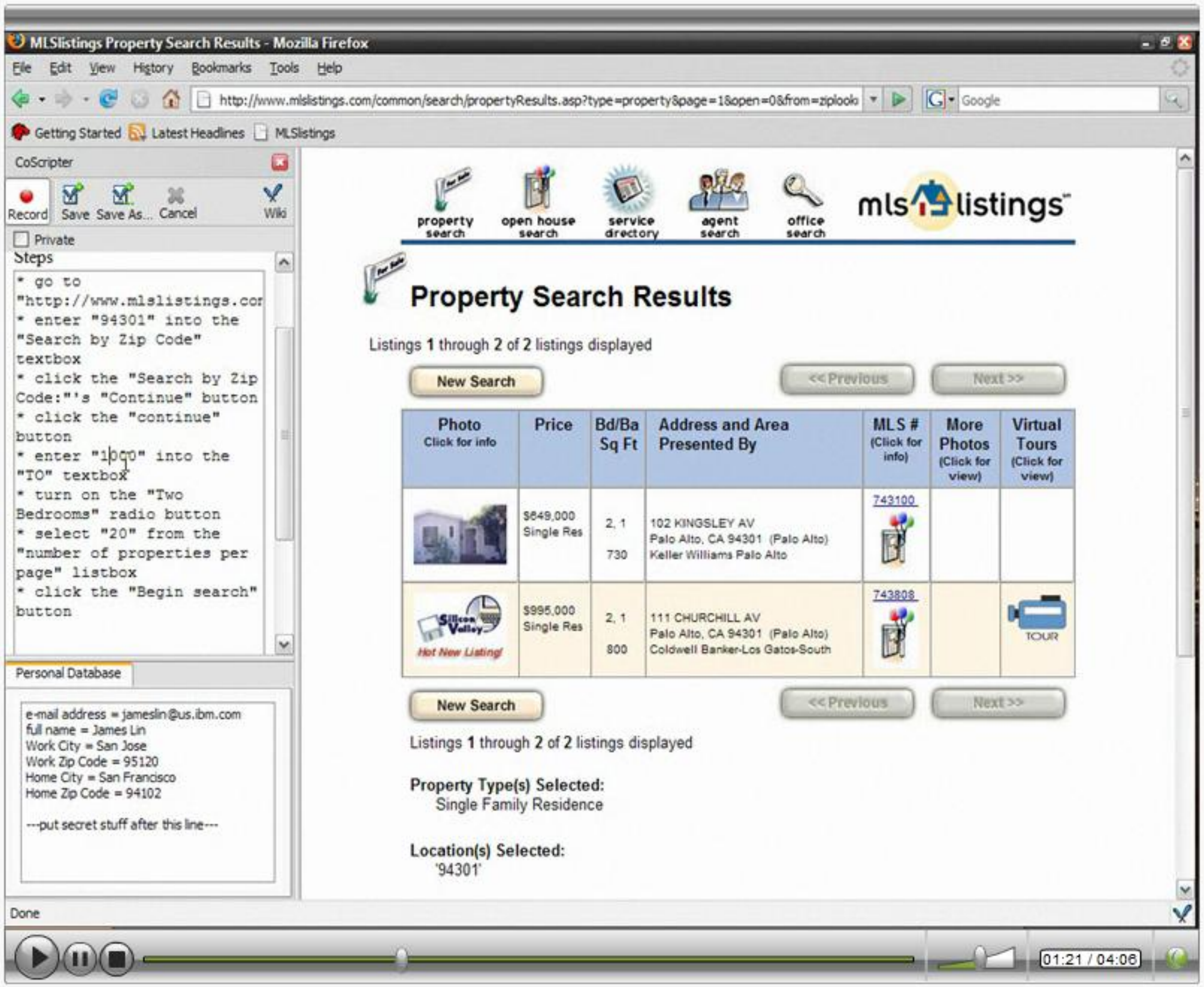
- Examples include…



Example: Autonomous UAV



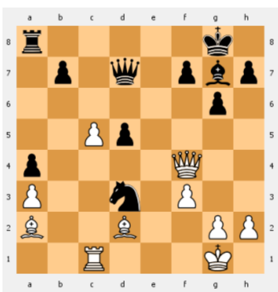
Example: Web Tasks

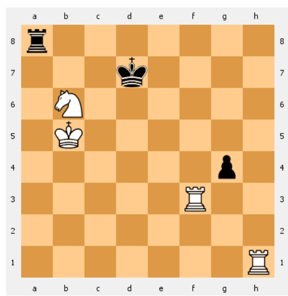


Motivation

- Most learning algorithms consume only positive and negative examples

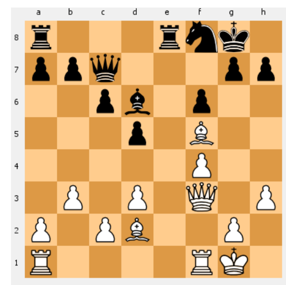
Positive

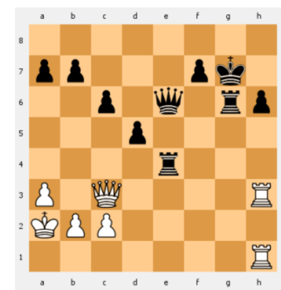


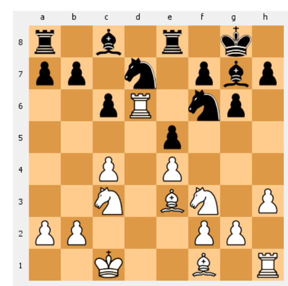


Concept
Knight-Fork

Negative

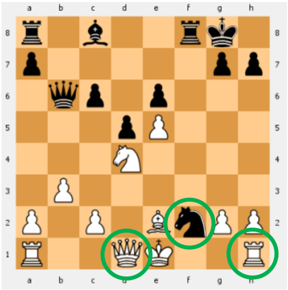




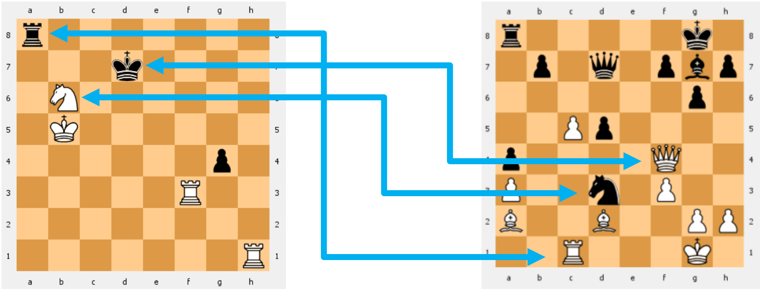


Information about Objects

- **Relevant objects**
  - "Which objects are relevant in this example?"



- **Object pairings**
  - "Given these examples, which objects have the same 'role'?"

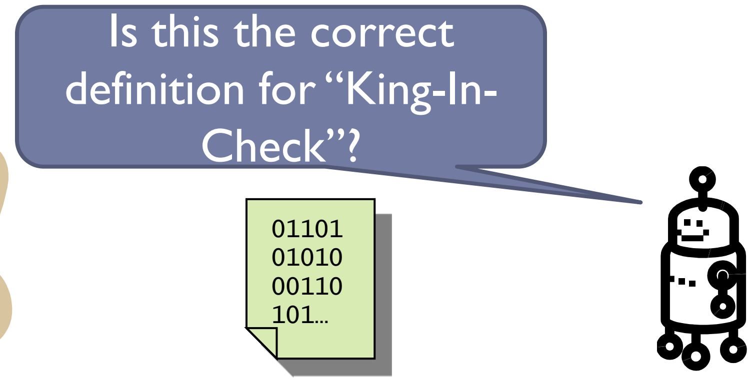


Framework: Exact learning from queries

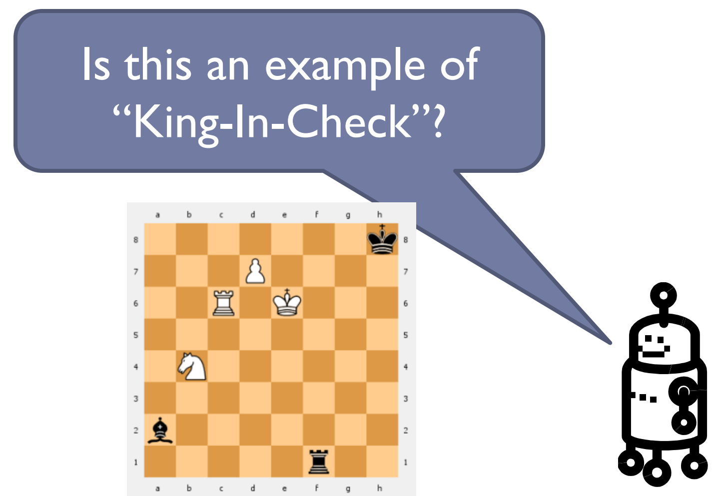
- Exactly learn first-order definite theories
  - ```
    mother(X,Y), father(Y,Z) -> grandfather(X,Z)
    father(X,Y), father(Y,Z) -> grandfather(X,Z)
    ```
- Learn via various query types (introduce later)
  - Equivalence Queries
  - Membership Queries
  - Relevant Object Queries
  - Pairing Queries
- Goal: Quantify number of queries to exactly learn definite theory
  - In particular, can object-based queries help?
    ```

Queries from Previous Work

- Equivalence Queries (EQs)
  - Source of new examples
  - Stopping point



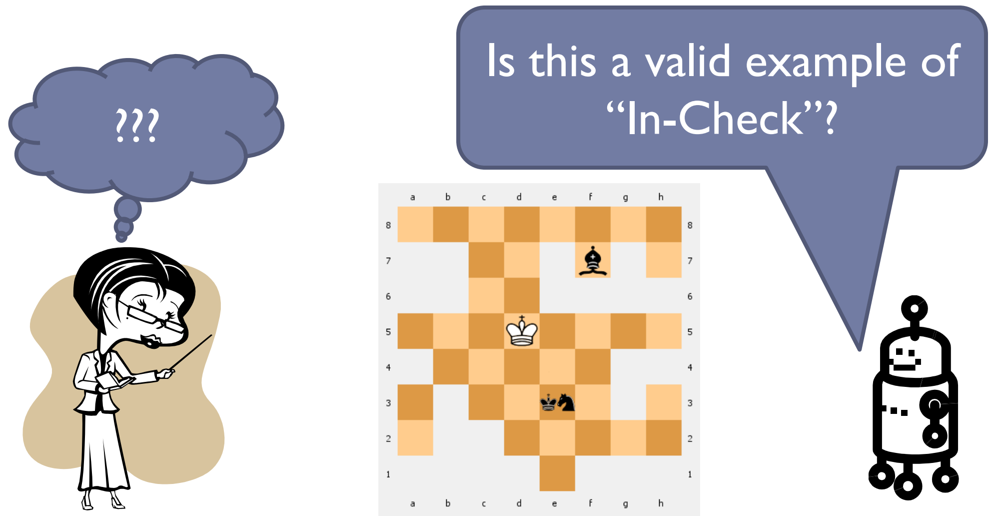
- Membership Queries (MQs)
  - Source of example labelings



- Prior work:
  - Angluin et al 1992
  - Reddy and Tadepalli 1997
  - Khardon 1999

Problems with MQs

- Teacher effort required can be high
- Algorithm may present nonsensical examples
- Small amount of information



- Can we reduce the number of MQs using object-based queries?

Learning with Relevant Object Queries

- Definition: A **relevant object query** (ROQ) takes a positive example as input and returns a minimal set of objects bound in a substitution for some clause in the target hypothesis.
- How best to leverage this information?

Algorithm schema

repeat

  if *EQ returns done* **then return** *the hypothesis H*
  Get a new counter-example from the previous EQ.
  Minimize the example by removing unnecessary objects
  Merge the example (if possible) into remembered examples
  Generate a new hypothesis H.

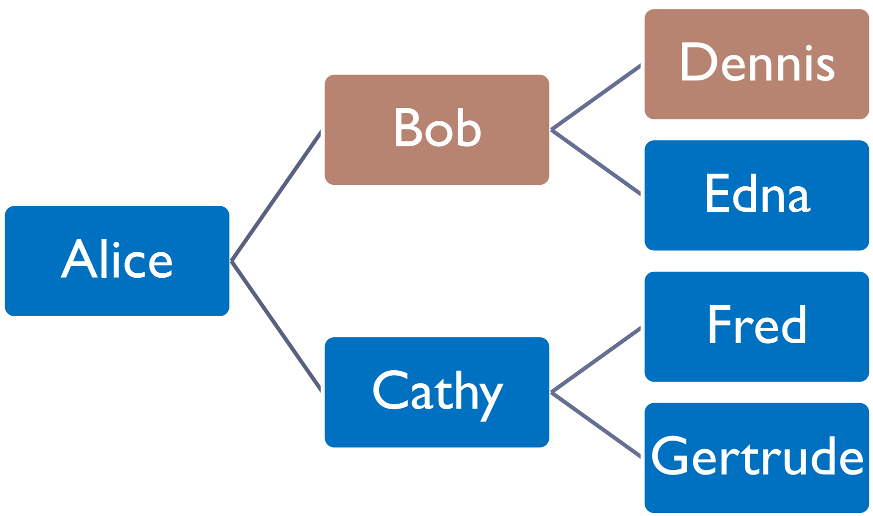
end

Algorithm Learn-MQ: Learns using EQ and MQ queries.

Algorithm schema: **Minimization**

- Example family tree with node color representing eye color
- FOL representation:

```
blue(a), brown(b), blue(c), brown(d), blue(e),
blue(f), blue(g), mother(a,c), father(a,b),
mother(b,e), father(b,d), mother(c,g), father(c,f)
  -> grandfather(a,d)
```

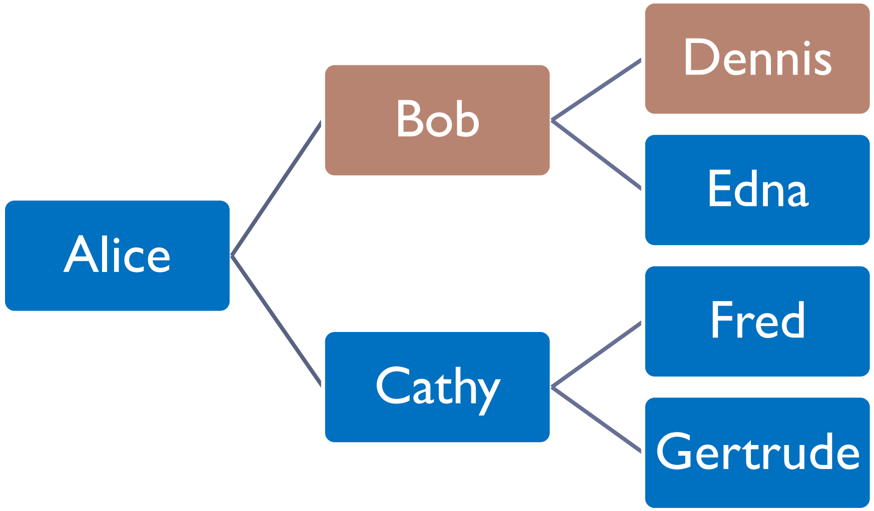



```
grandfather(X,Z) :=
  mother(X,Y),
  father(Y,Z).

grandfather(X,Z) :=
  father(X,Y),
  father(Y,Z).
```

Algorithm schema: **Minimization**

- FOL representation:
  - `blue(a), brown(b), blue(c), brown(d), blue(e), blue(f), blue(g), mother(a,c), father(a,b), mother(b,e), father(b,d), mother(c,g), father(c,f) -> grandfather(a,d)`

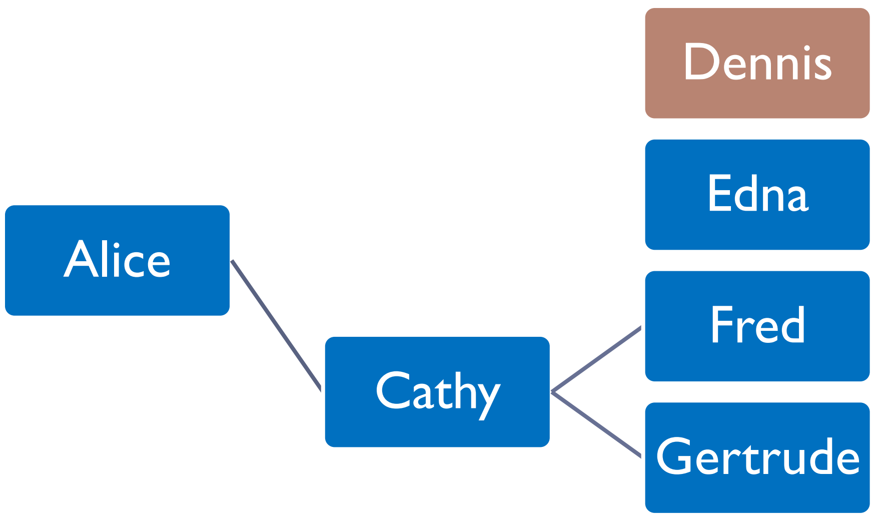



```
grandfather(X,Z) :=
  mother(X,Y),
  father(Y,Z).

grandfather(X,Z) :=
  father(X,Y),
  father(Y,Z).
```

Algorithm schema: **Minimization**

- FOL representation:
  - `blue(a), blue(c), brown(d), blue(e), blue(f), blue(g), mother(a,c), mother(c,g), father(c,f)`
    `-> grandfather(a,d)`



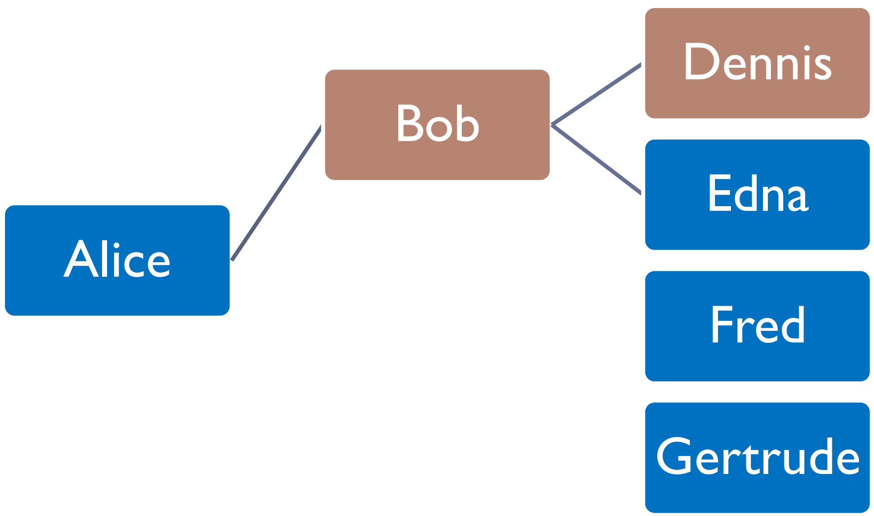

```
grandfather(X,Z) :=
  mother(X,Y),
  father(Y,Z).

grandfather(X,Z) :=
  father(X,Y),
  father(Y,Z).
```

Result of MQ on above example: FALSE

Algorithm schema: **Minimization**

- FOL representation:
  - `blue(a), brown(b), brown(d), blue(e), blue(f), blue(g), father(a,b), mother(b,e), father(b,d)`
    `-> grandfather(a,d)`



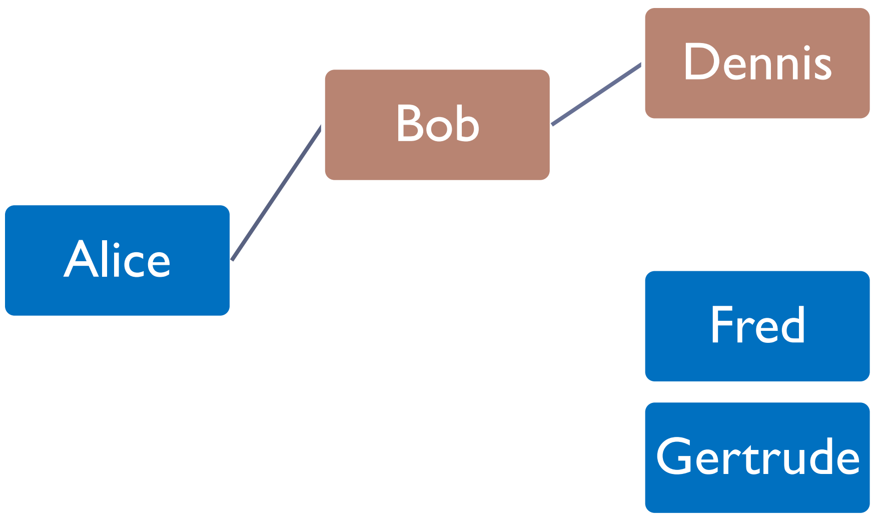

```
grandfather(X,Z) :=
  mother(X,Y),
  father(Y,Z).

grandfather(X,Z) :=
  father(X,Y),
  father(Y,Z).
```

Result of MQ on above example: True

Algorithm schema: **Minimization**

- FOL representation:
  - ```
    blue(a), brown(b), brown(d), blue(f), blue(g),
    father(a,b), father(b,d)
      -> grandfather(a,d)
    ```



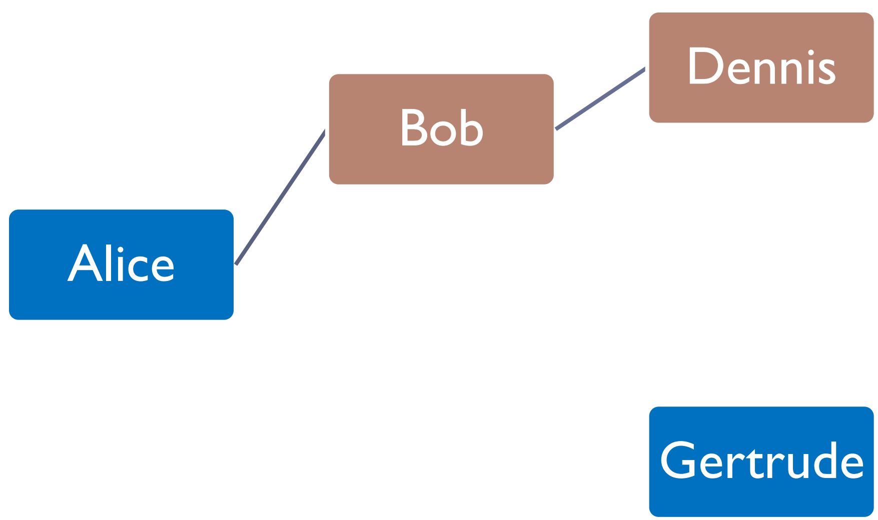

```
grandfather(X,Z) :=
  mother(X,Y),
  father(Y,Z).

grandfather(X,Z) :=
  father(X,Y),
  father(Y,Z).
```

Result of MQ on above example: True
```

Algorithm schema: **Minimization**

- FOL representation:
  - `blue(a), brown(b), brown(d), blue(g), father(a,b), father(b,d)`
    `-> grandfather(a,d)`



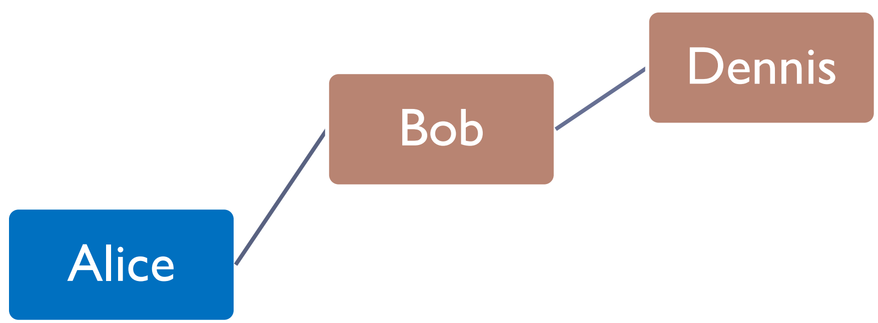

```
grandfather(X,Z) :=
  mother(X,Y),
  father(Y,Z).

grandfather(X,Z) :=
  father(X,Y),
  father(Y,Z).
```

Result of MQ on above example: True

Algorithm schema: **Minimization**

- FOL representation:
  - `blue(a), brown(b), brown(d), father(a,b), father(b,d) -> grandfather(a,d)`



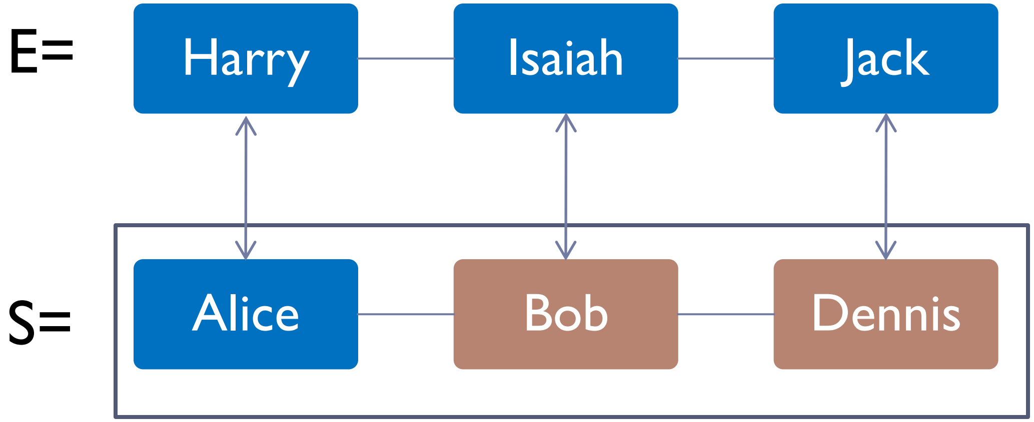

```
grandfather(X,Z) :=
  mother(X,Y),
  father(Y,Z).

grandfather(X,Z) :=
  father(X,Y),
  father(Y,Z).
```

Result of MQ on above example: True

Algorithm Schema: **Merging**

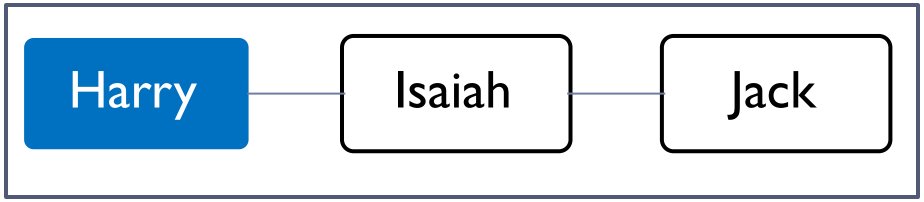
- ```
  E = grandfather(h,j) :=
          blue(h), blue(i), blue(j),
          father(h,i), father(i,j).
  ```
- ```
  S = grandfather(a,d) :=
          blue(a), brown(b), brown(d),
          father(a,b), father(b,d).
  ```



Algorithm Schema: **Merging**

- ```
  S1' = grandfather(h,j) :=
            blue(h), father(a,b),
            father(b,d).
  ```
- MQ(S1') returns true

S=


  ```

Result 1 (Learn-MQ-ROQ)

- n = max # of objects in an example
- k = max # of variables in a clause in target clause
- a = max arity of a predicate

- Compared to (Khardon 1999), Learn-MQ-ROQ reduces the number of MQs by $O(nk^a)$ by introducing $O(k^a)$ ROQs
  - # of queries no longer depends on n

- MQs are still used in merging step
  - Can we completely eliminate MQs?

Negatively-Biased EQ Oracle

- An oracle is **negatively-biased** if it answers equivalence queries by always providing a negative counter-example if one exists
  - If none exists, returns a positive counter-example (or halt)
- Embeds the merging test in the EQ
- Can be simulated given a large number of negative examples
  - Or large set of unlabeled examples that are mostly negative

Result 2 (Learn-ROQ)

- Using a negatively-biased EQ oracle and ROQs, Learn-ROQ is efficient and eliminates MQs.
  - Now requires an EQ for every MQ used during merging

- May be beneficial in situations with a large number of mostly negative examples

Inexact ROQs

- Allow an oracle that answers ROQs to error (in a restricted way)

- An oracle is ***(j,f)*-verbose** if for *j* of the clauses it marks *f* extra objects as relevant

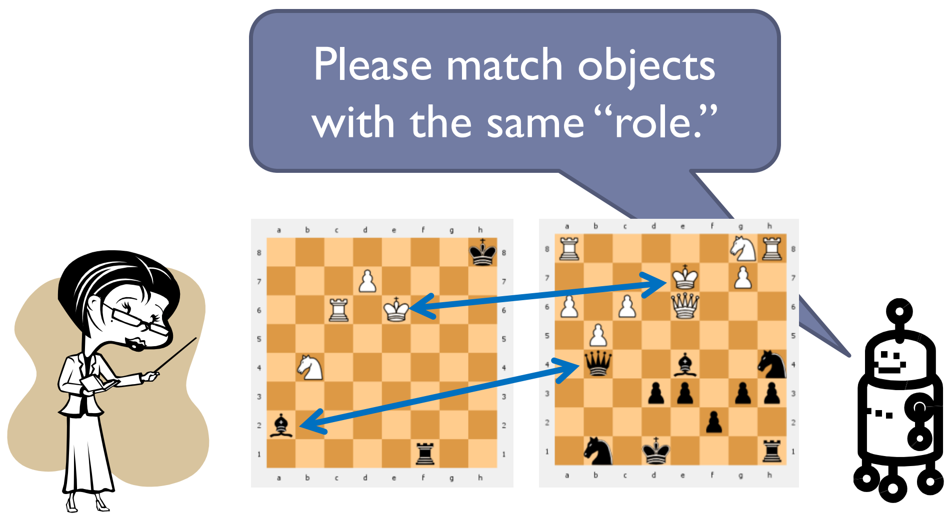
- Similarly, an oracle is ***(j,f)*-conservative** if for *j* of the clauses it misses *f* of the relevant objects

Result 3 (inexact ROQs)

- (j,f)-verbose oracle
  - Can use previous algorithms directly
  - Increase of O((k+f)^a) EQs and ROQs!
- (j,f)-conservative oracle
  - Algorithm appears to require MQs
  - Adds at most n-(k+f) MQs for every error
- Above results suggest a conservative oracle is preferable when MQs are available

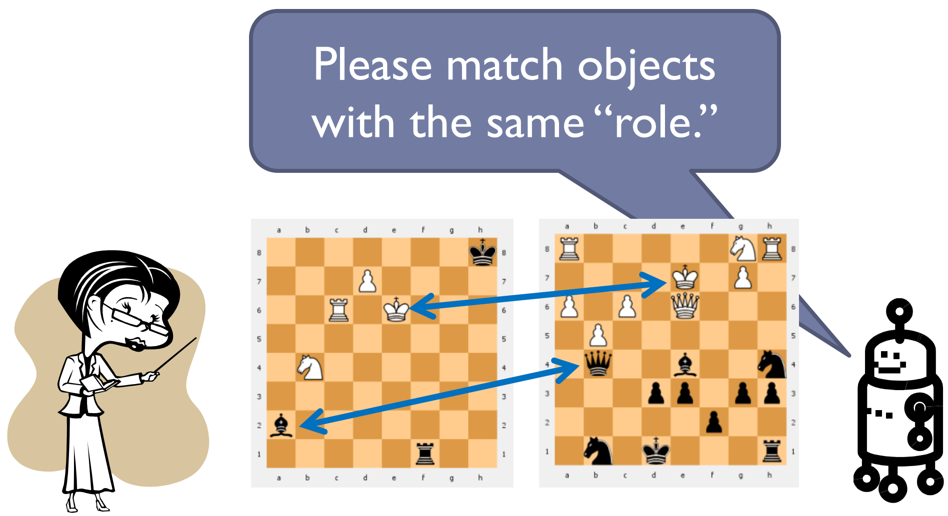
Learning with Pairing Queries

- Definition: A **pairing query** (PQ) is a query that, given two positive examples, returns false if there is no clause in T that covers them both. Otherwise, a 1-1 mapping between objects in E1 and E2 is returned where objects are mapped together if they correspond to the same variable in T.



Result 4 (pairing queries)

- Using PQs, we can learn with no MQs or ROQs and O(k^a) PQs.
- Likely not easy to answer in practice
- Inexact pairing queries?



Future work

- User study
- Probabilistic model of oracle mistakes
- New query types

Questions?