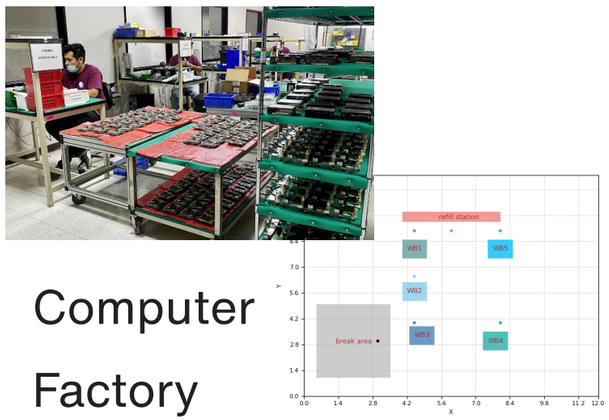
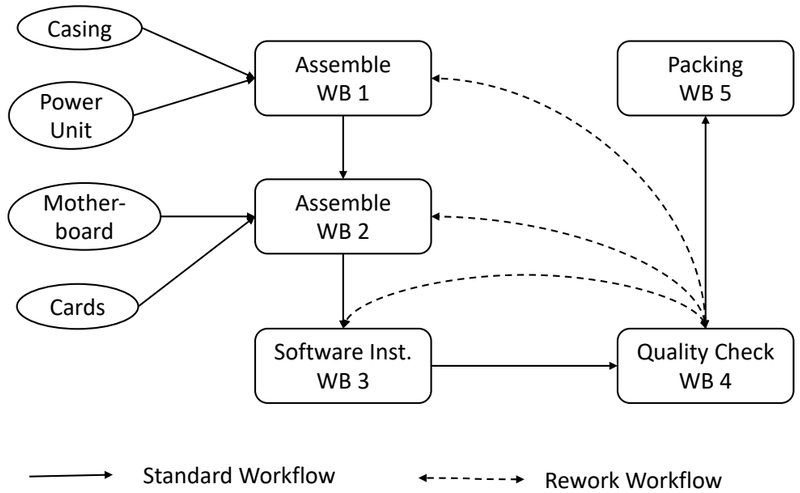
Bottom-Up Stratified Probabilistic Logic Programming with Fusemate

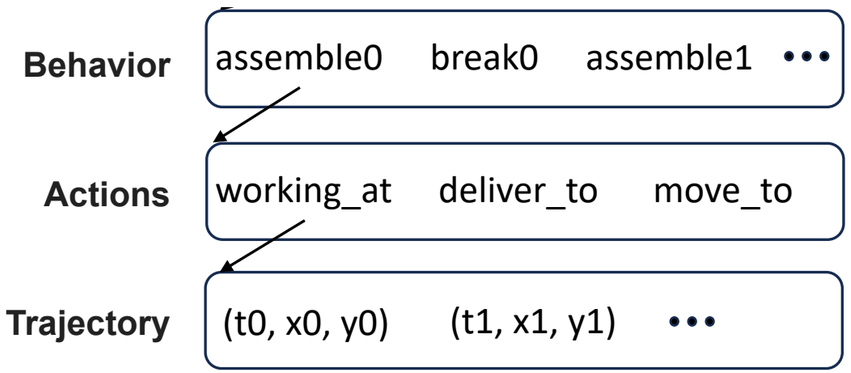
Peter Baumgartner and Elena Tartaglia
Data61/CSIRO

Background: Movement Analytics for Productivity, Efficiency and Safety



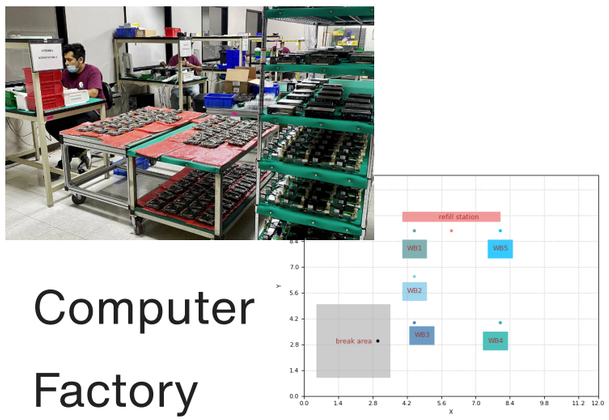
Computer Factory



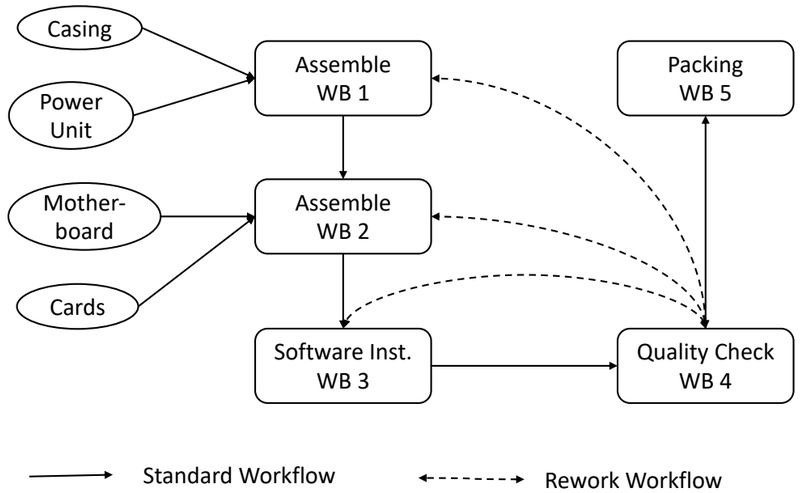


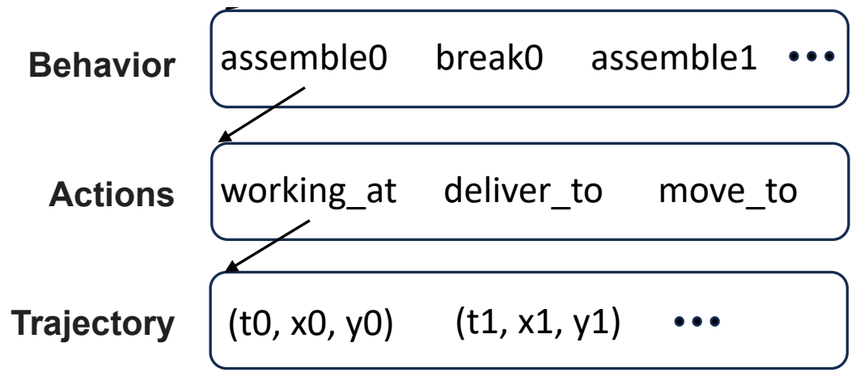
Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

Background: Movement Analytics for Productivity, Efficiency and Safety



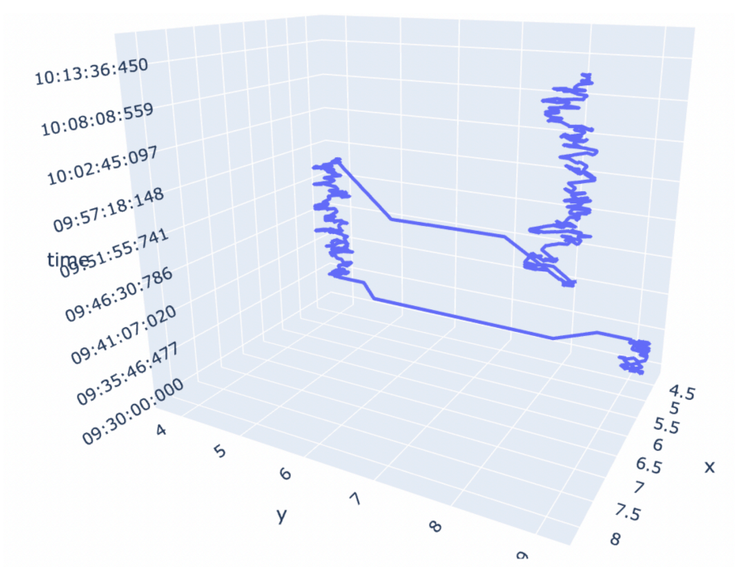
Computer Factory



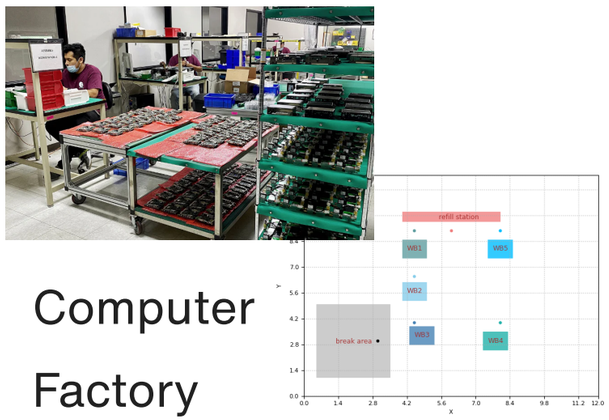


Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

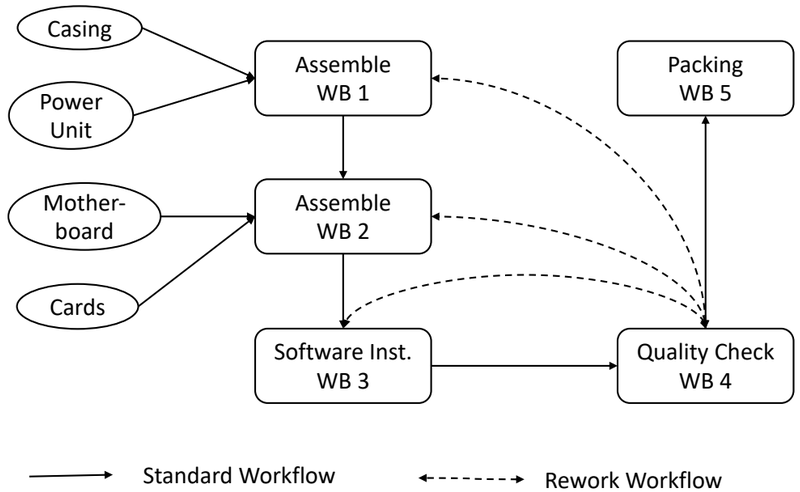
Given trajectory

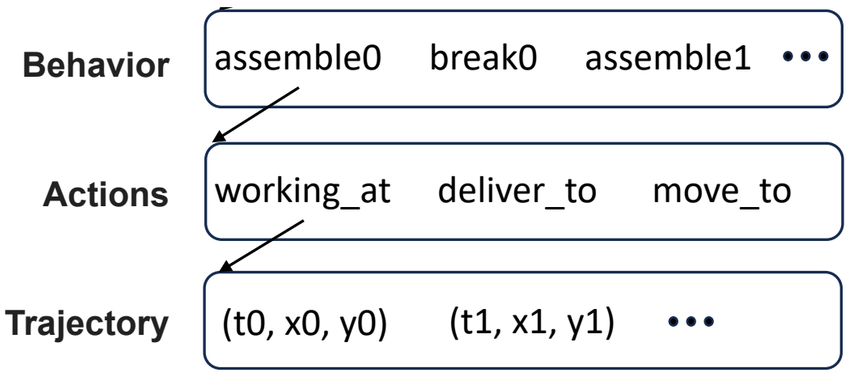


Background: Movement Analytics for Productivity, Efficiency and Safety



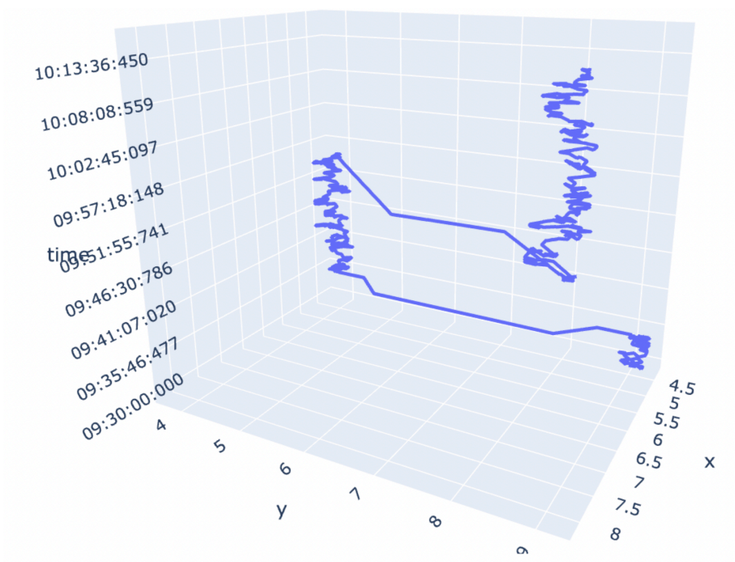
Computer Factory





Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

Given trajectory

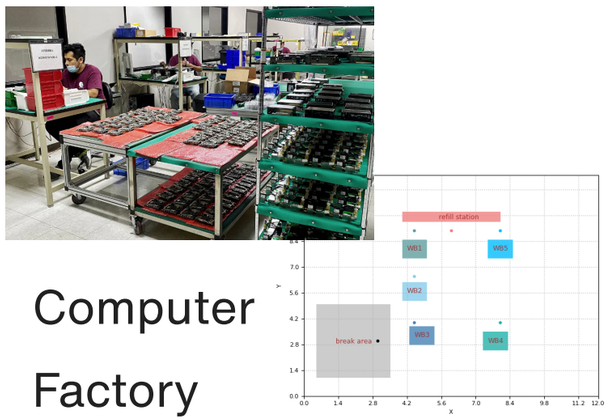


Probabilistic logic program

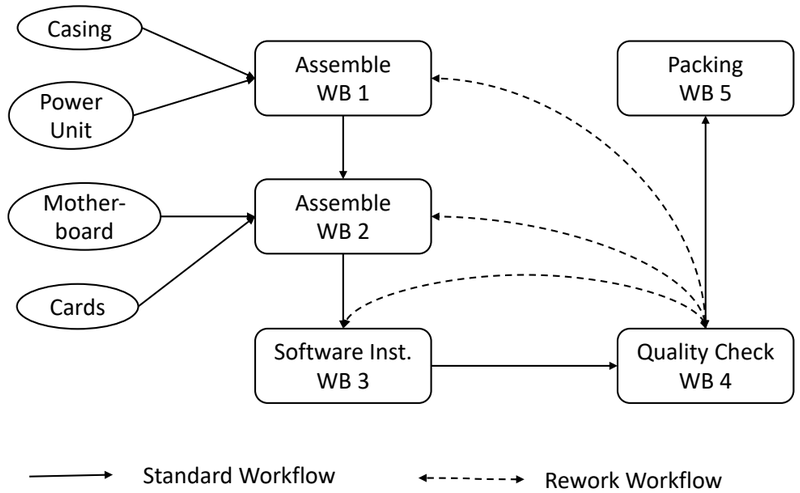
(MLE, Hidden Markov Model, Viterbi Alg)

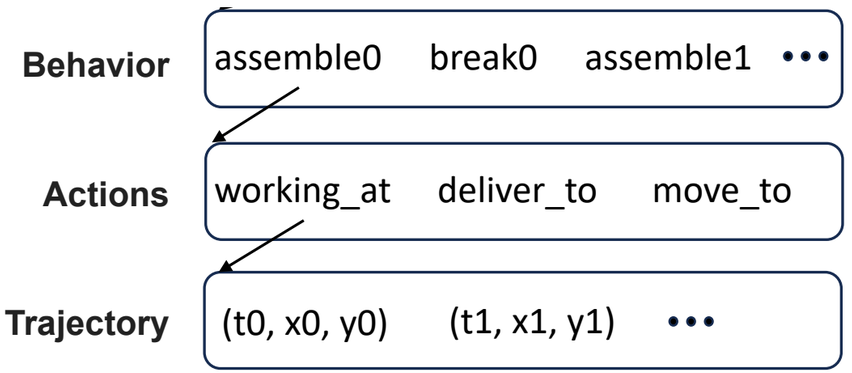
```
behaviour ~ [assemble, break ...]. %% Distribution
worker ~ [1,2,3,4,5].              %% Distribution

action = working_at(wb(W))   @ 0 :-
    behaviour = assemble,
    worker = W.

action = deliver_to(wb(W+1)) @ 1 :-
    behaviour = assemble,
    worker = W.

loc = L       @ T :- action = working_at(L) @ T.
dur ~ [1..10] @ T :- action = working_at(_) @ T.
```

Background: Movement Analytics for Productivity, Efficiency and Safety



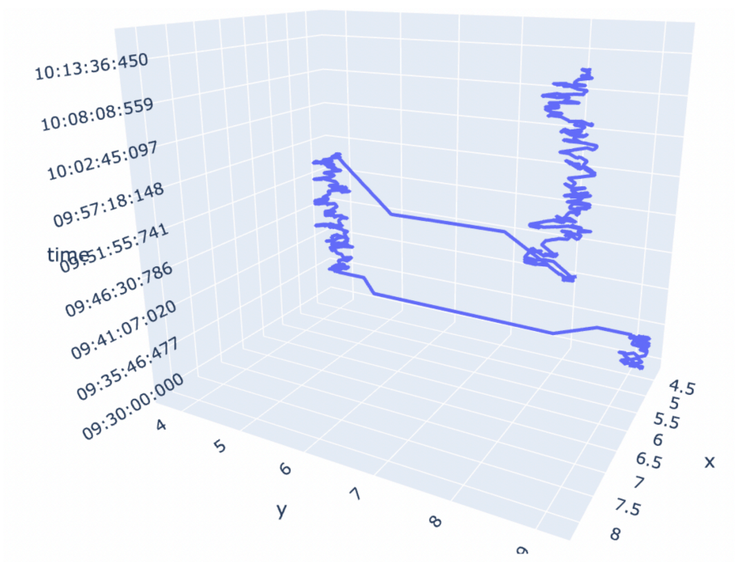
Computer Factory





Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

Given trajectory



Probabilistic logic program

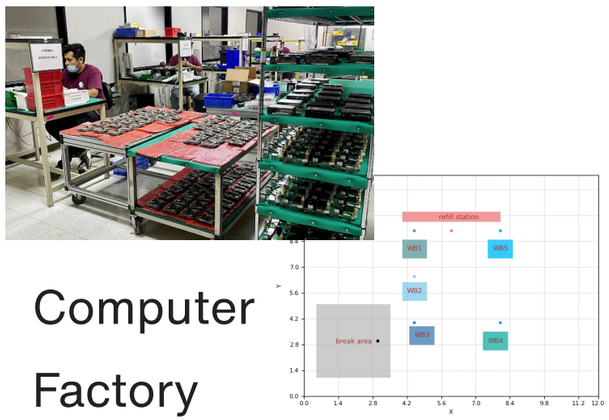
(MLE, Hidden Markov Model, Viterbi Alg)

```
behaviour ~ [assemble, break ...]. %% Distribution
worker ~ [1,2,3,4,5].              %% Distribution

action = working_at(wb(W))   @ 0 :-
    behaviour = assemble,
    worker = W.

action = deliver_to(wb(W+1)) @ 1 :-
    behaviour = assemble,
    worker = W.

loc = L       @ T :- action = working_at(L) @ T.
dur ~ [1..10] @ T :- action = working_at(_) @ T.
```

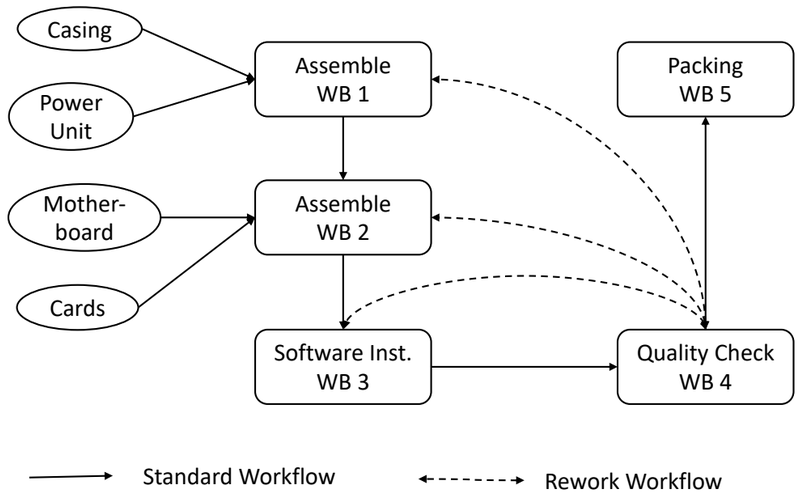
Most likely behaviour seq.

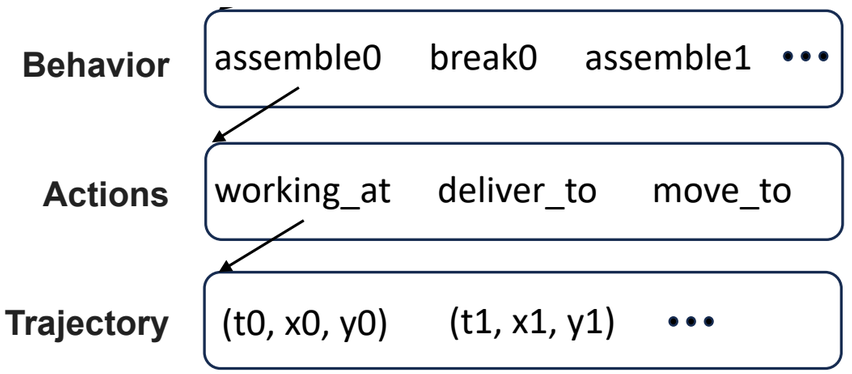
assemble -> break -> ...

Background: Movement Analytics for Productivity, Efficiency and Safety



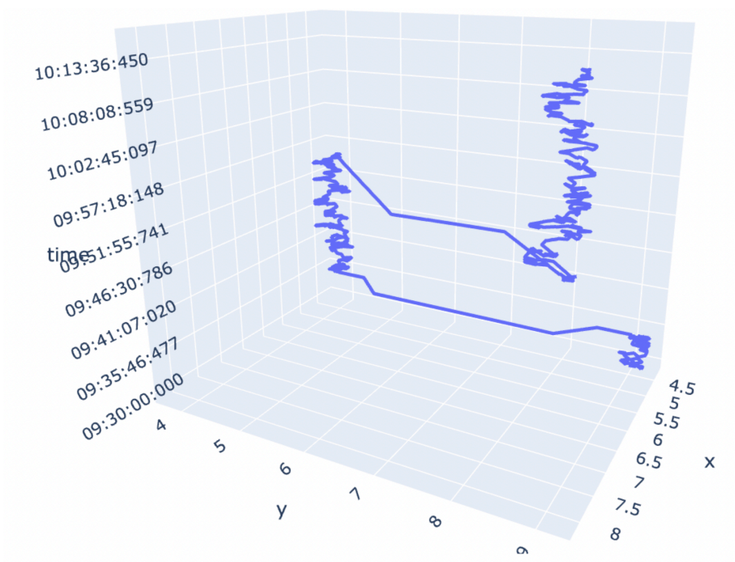
Computer Factory





Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

Given trajectory



Probabilistic logic program

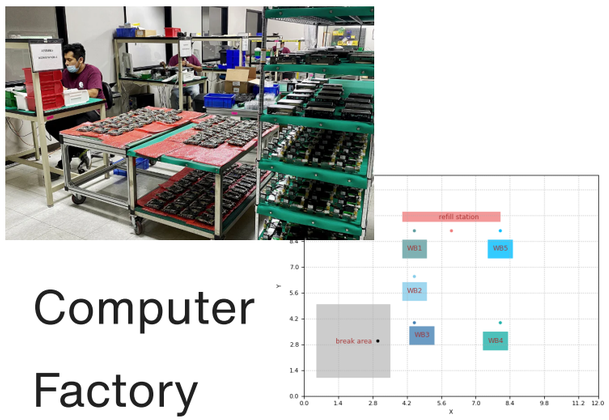
(MLE, Hidden Markov Model, Viterbi Alg)

```
behaviour ~ [assemble, break ...]. %% Distribution
worker ~ [1,2,3,4,5].              %% Distribution

action = working_at(wb(W))    @ 0 :-
    behaviour = assemble,
    worker = W.

action = deliver_to(wb(W+1)) @ 1 :-
    behaviour = assemble,
    worker = W.

loc = L       @ T :- action = working_at(L) @ T.
dur ~ [1..10] @ T :- action = working_at(_) @ T.
```

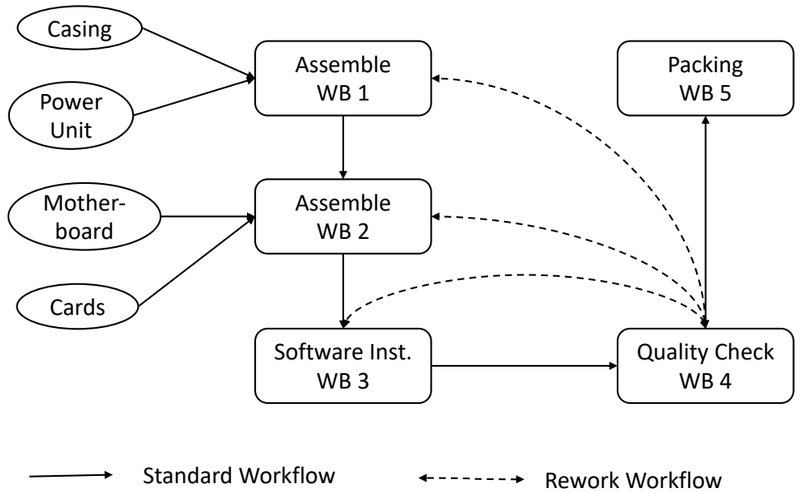
Most likely behaviour seq.

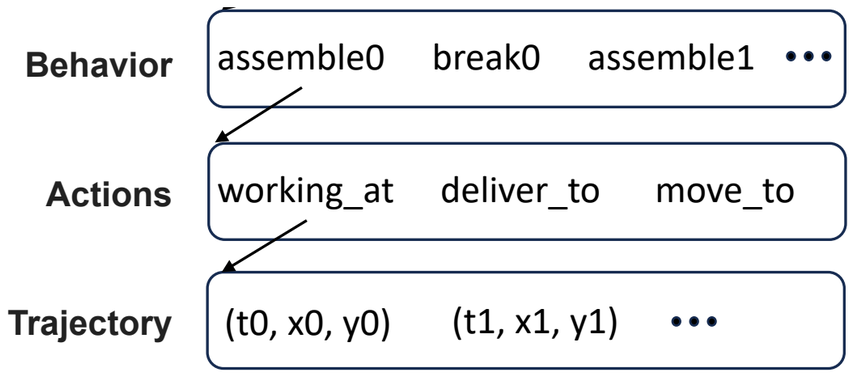
assemble -> break -> ...

Background: Movement Analytics for Productivity, Efficiency and Safety



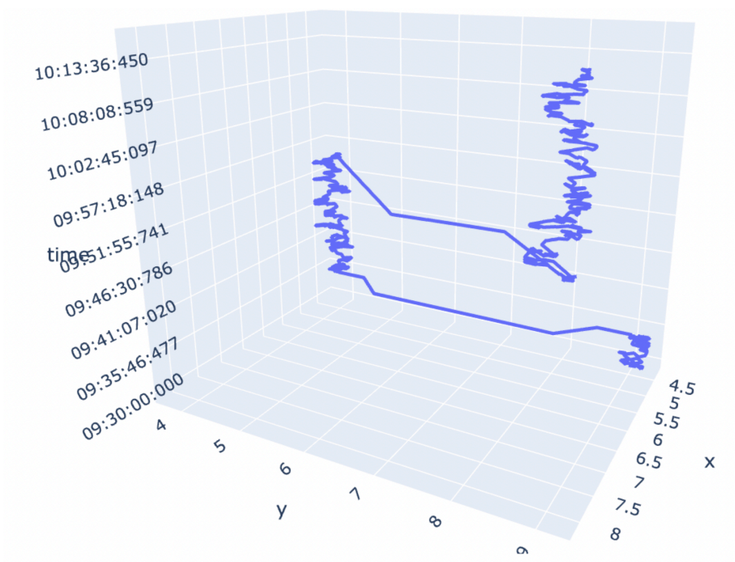
Computer Factory





Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

Given trajectory



Probabilistic logic program

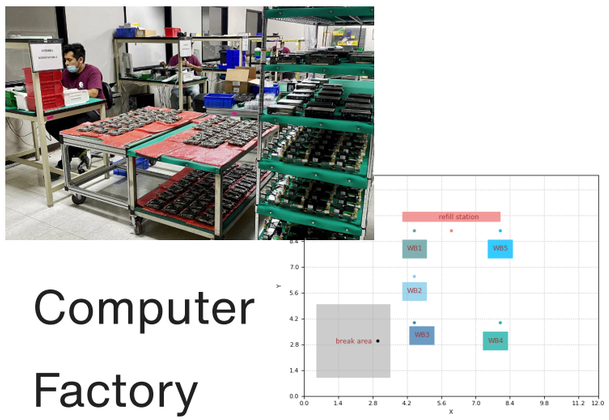
(MLE, Hidden Markov Model, Viterbi Alg)

```
behaviour ~ [assemble, break ...]. %% Distribution
worker ~ [1,2,3,4,5].              %% Distribution

action = working_at(wb(W))   @ 0 :-
    behaviour = assemble,
    worker = W.

action = deliver_to(wb(W+1)) @ 1 :-
    behaviour = assemble,
    worker = W.

loc = L       @ T :- action = working_at(L) @ T.
dur ~ [1..10] @ T :- action = working_at(_) @ T.
```

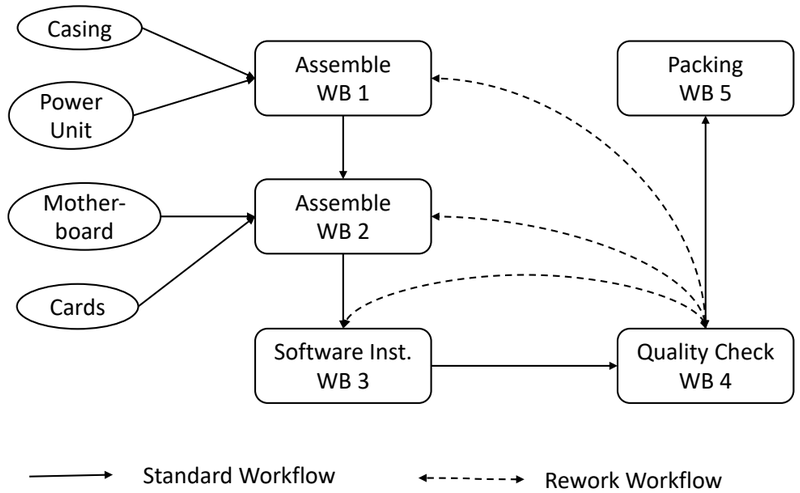
Most likely behaviour seq.

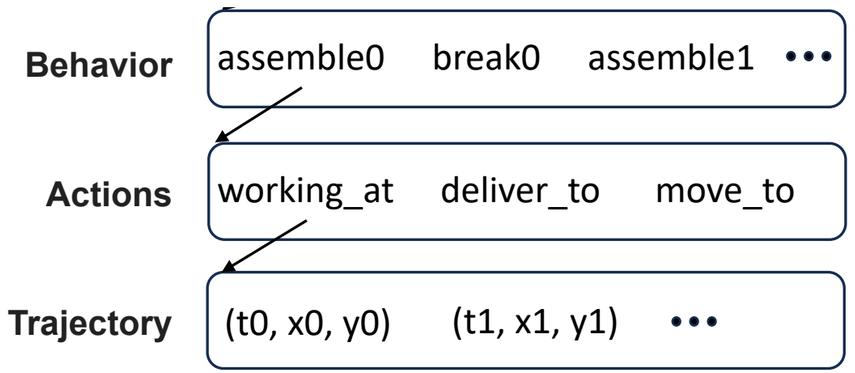
assemble -> break -> ...

Background: Movement Analytics for Productivity, Efficiency and Safety



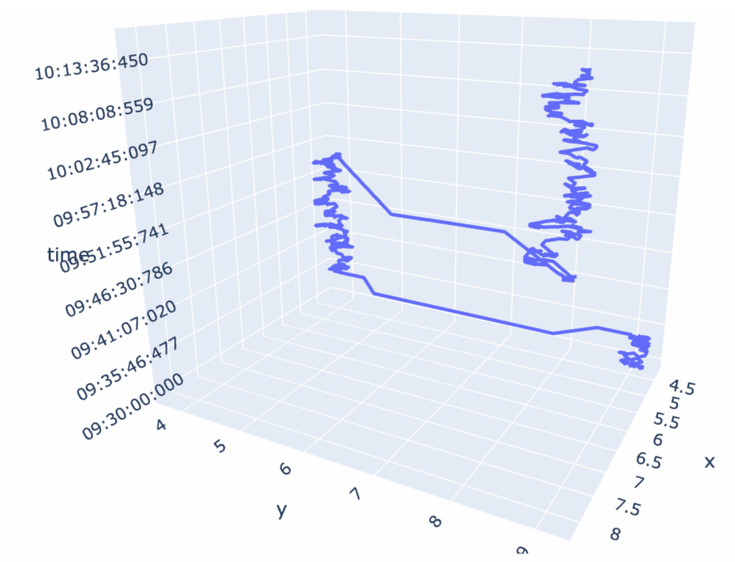
Computer Factory





Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

Given trajectory



Probabilistic logic program

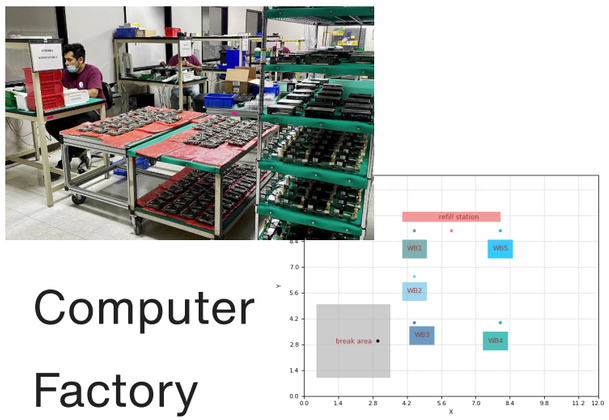
(MLE, Hidden Markov Model, Viterbi Alg)

```
behaviour ~ [assemble, break ...].  %% Distribution
worker ~ [1,2,3,4,5].               %% Distribution

action = working_at(wb(W))    @ 0 :-
    behaviour = assemble,
    worker = W.

action = deliver_to(wb(W+1)) @ 1 :-
    behaviour = assemble,
    worker = W.

loc = L       @ T :- action = working_at(L) @ T.
dur ~ [1..10] @ T :- action = working_at(_) @ T.
```

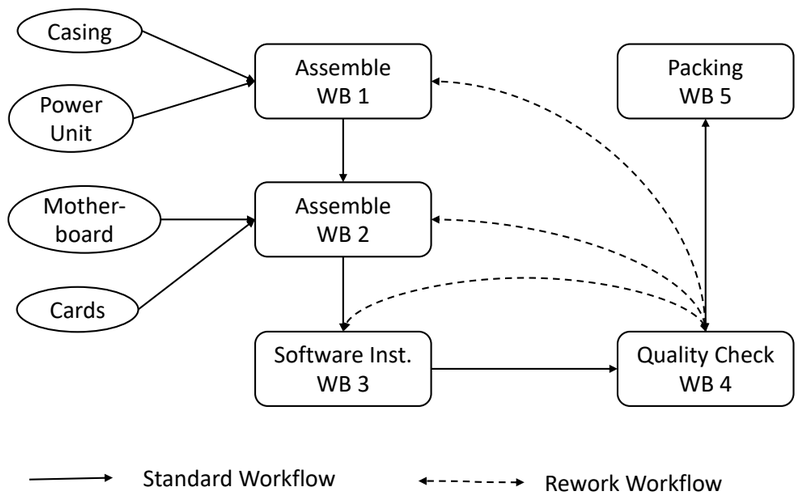
Most likely behaviour seq.

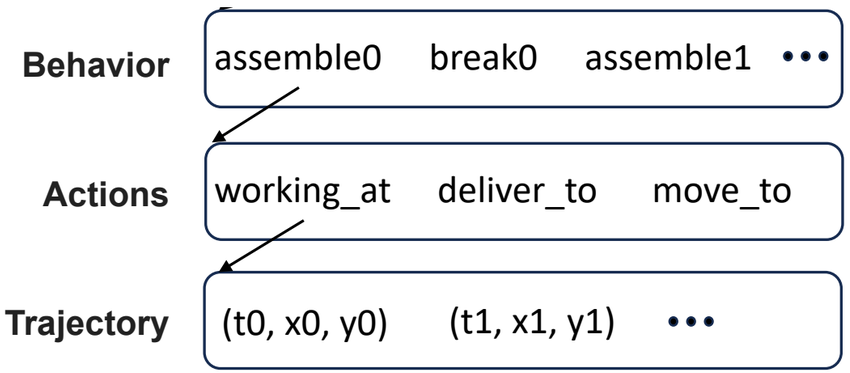
assemble -> break -> ...

Background: Movement Analytics for Productivity, Efficiency and Safety



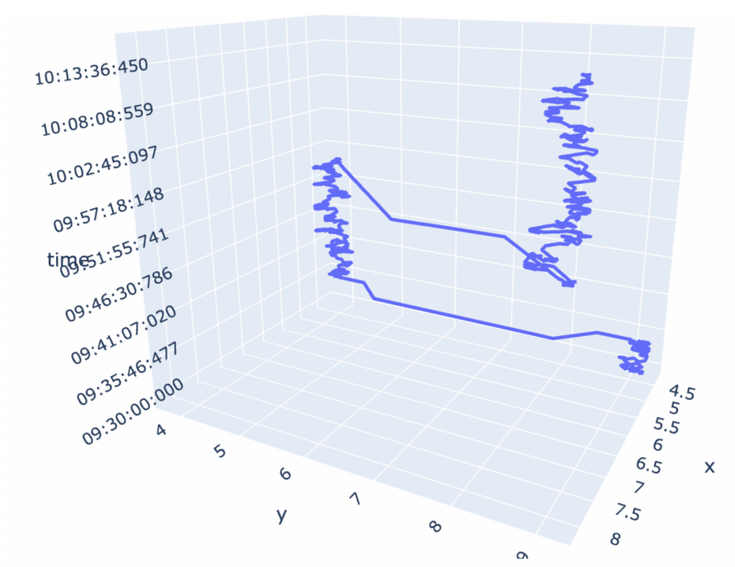
Computer Factory





Problem: Trajectory classification: what actions/behaviours exhibited by a trajectory?

Given trajectory



Probabilistic logic program

(MLE, Hidden Markov Model, Viterbi Alg)

```
behaviour ~ [assemble, break ...].  %% Distribution
worker ~ [1,2,3,4,5].               %% Distribution

action = working_at(wb(W))    @ 0 :-
    behaviour = assemble,
    worker = W.

action = deliver_to(wb(W+1)) @ 1 :-
    behaviour = assemble,
    worker = W.

loc = L       @ T :- action = working_at(L) @ T.
dur ~ [1..10] @ T :- action = working_at(_) @ T.
```

Most likely behaviour seq.

assemble -> break -> ...

Challenge: Search Space

Paper: our solution

This Paper

A probabilistic logic programming language

- Probabilistic annotated heads
- Discrete distributions
- Built-in semantics of equations
- Discrete time
- Stratified default negation “by predicates and time”

```
0.98 :: happy(X) :- has_ICLP_paper(X).
X ~ [1..6] :- fair_dice(X).
1/6 :: X = 1 + ... + 1/6 :: X = 6 :- fair_dice(X).
x = 5 and x = 6 are inconsistent
p @ T+1 :- p @ T.
p @ T :- q @ T,
    \+ r @ T,
    \+ p @ T-1.
```

This Paper

A probabilistic logic programming language

- Probabilistic annotated heads
- Discrete distributions
- Built-in semantics of equations
- Discrete time
- Stratified default negation “by predicates and time”

```
0.98 :: happy(X) :- has_ICLP_paper(X).
X ~ [1..6] :- fair_dice(X).
1/6 :: X = 1 + ... + 1/6 :: X = 6 :- fair_dice(X).
x = 5 and x = 6 are inconsistent
p @ T+1 :- p @ T.
p @ T :- q @ T,
    \+ r @ T,
    \+ p @ T-1.
```

Two-phase probabilistic inference algorithm

- Phase 1: grounding the program; also removes default negation

  Phase 2: (stochastic) variable elimination on ground program

Main contribution: *efficient bottom-up* grounding algorithm

- *Query guidance*: query regression + inconsistency pruning
- Good experimental results for e.g. filtering queries

```
?- state=S @ 10 | obs = .. @ 0, ..., obs = .. @ 10.
```

Example Fusemate Probabilistic Logic Program

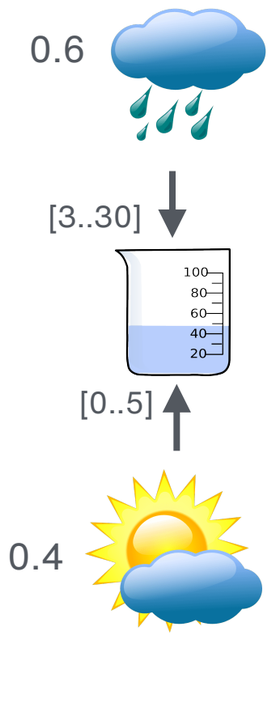
Drawing without replacement

```
urn([r(1), r(2), g(1)]) @ 0.                %% Initially two red and one green distinguishable balls
draw ~ Balls @ T :-                         %% Draw  a ball uniformly if urn is not empty
    urn(Balls) @ T,
    Balls \= [].
urn(Balls -- [B]) @ T+1 :-                  %% Drawing a ball removes it from urn
    urn(Balls) @ T,
    draw = B @ T.
some(red) @ T :- draw=r(_) @ T.             %% Abstract from ball id, color only
some(green) @ T :- draw=g(_) @ T.
```

Queries

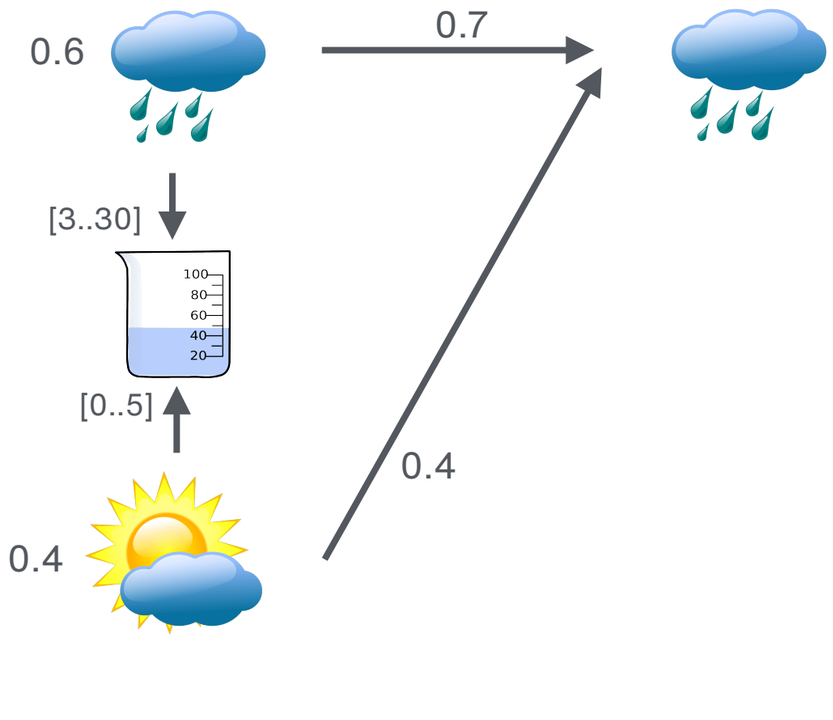
```
?- some(green) @ 0.
% 0.333333

?- some(green) @ 1 | some(red) @ 0.
% 0.5 conditional query

?- some(C1) @ 1, some(C2) @ 2 | some(red) @ 0. % Non-ground conditional query, two solutions:
% 0.5 :: [C1 = red, C2 = green]
% 0.5 :: [C1 = green, C2 = red]
```

Hidden Markov Model



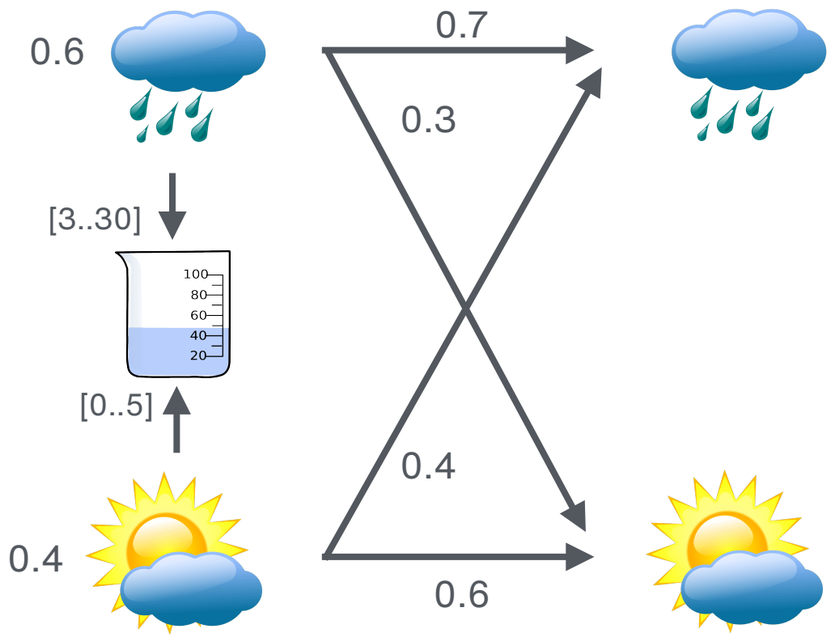
✔

Hidden Markov Model



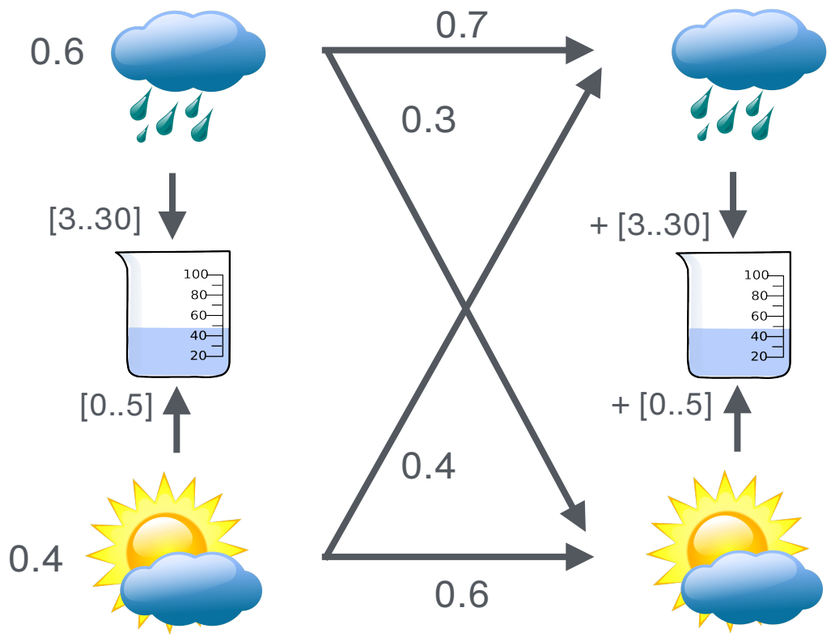
✔

Hidden Markov Model



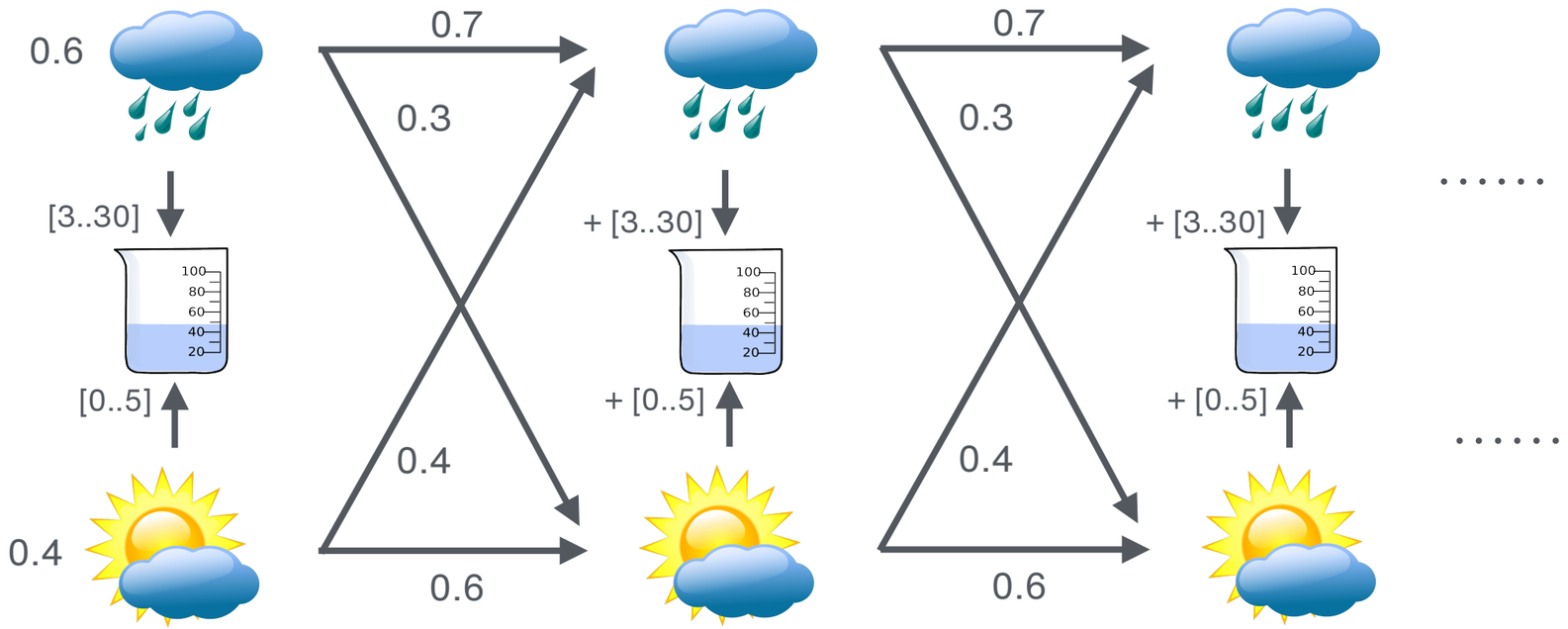
✔

Hidden Markov Model

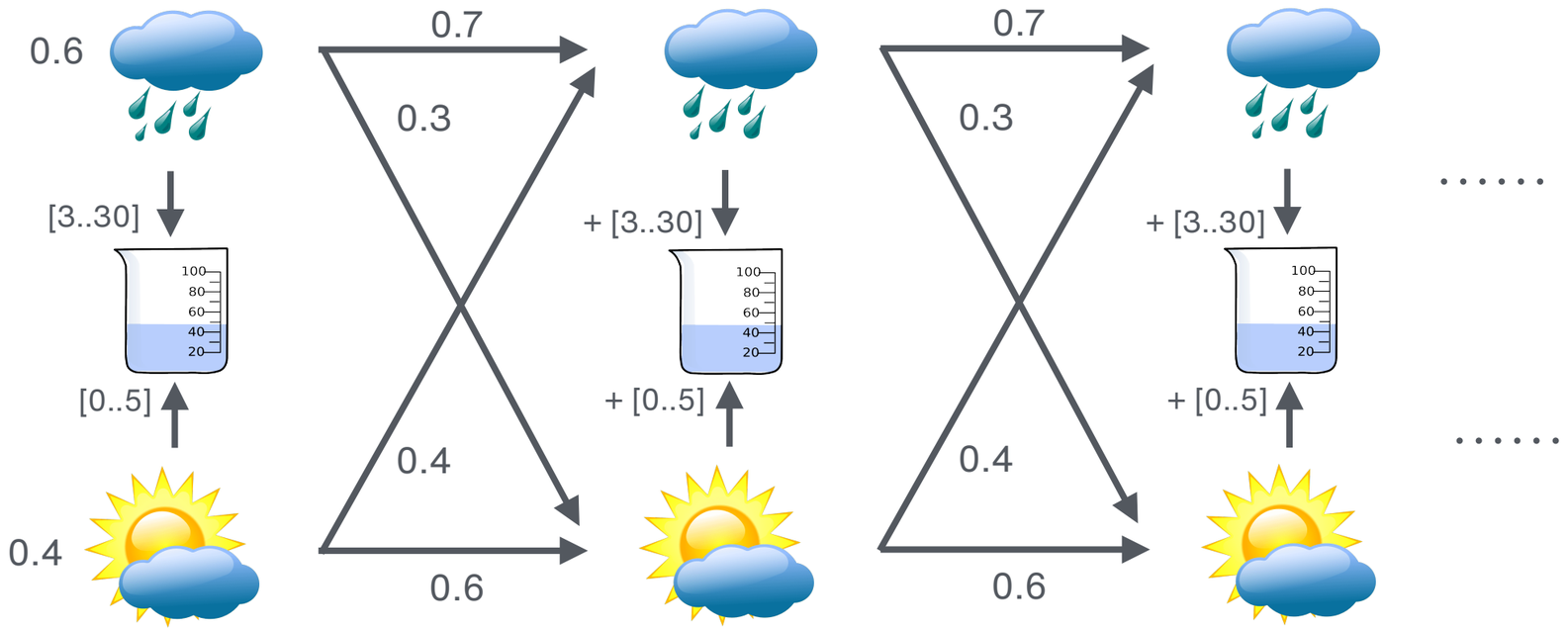


✔

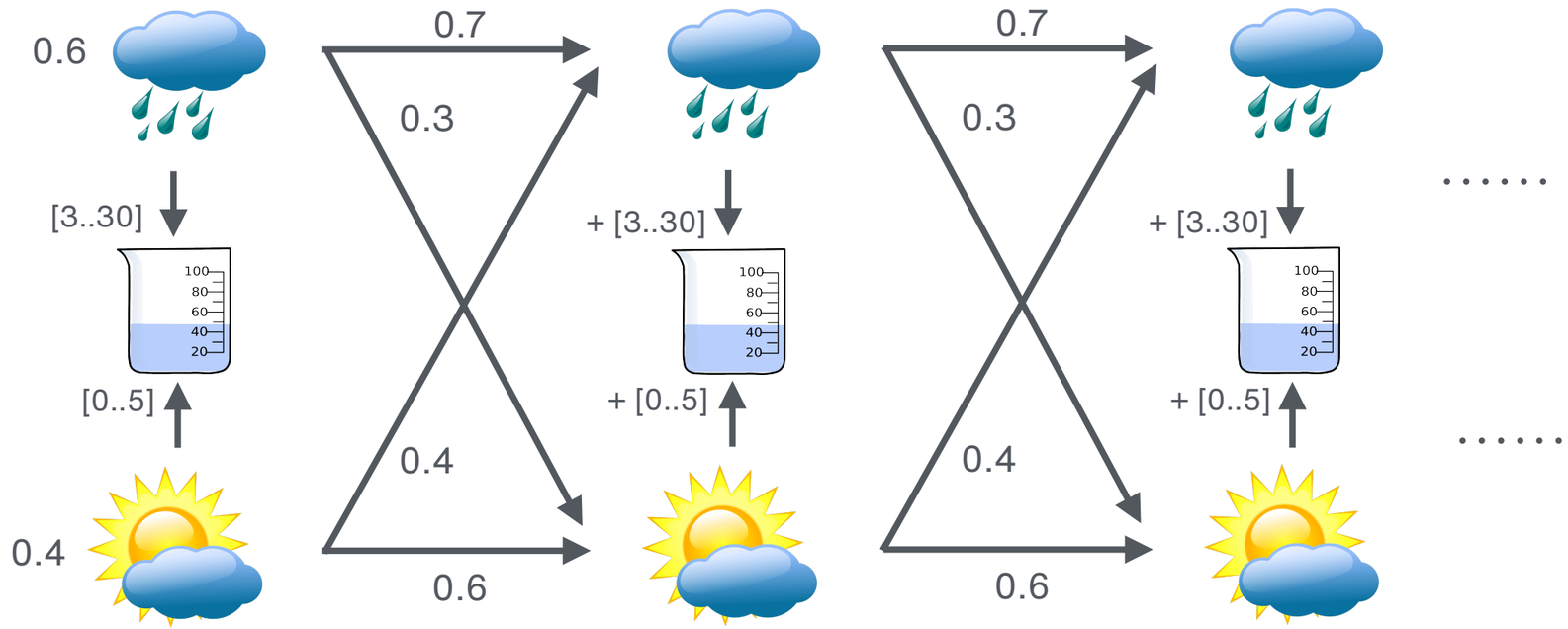
Hidden Markov Model



Hidden Markov Model



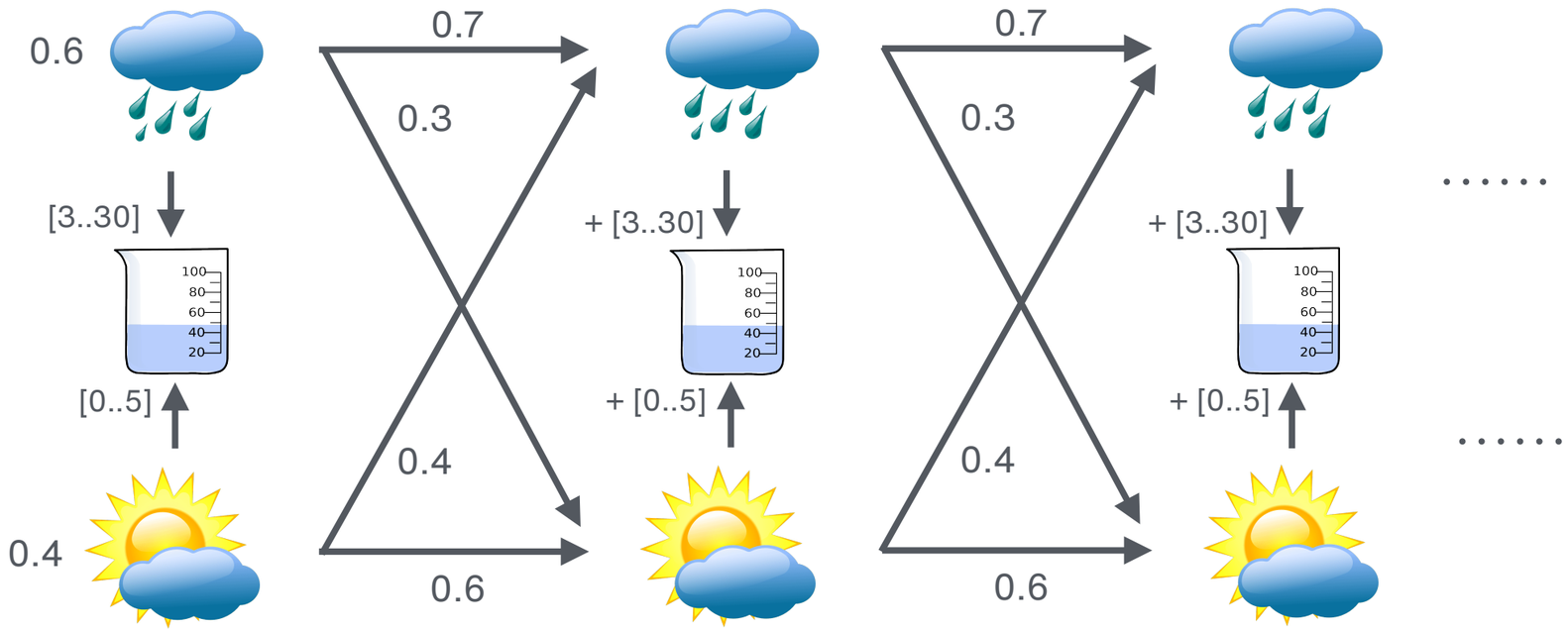
Hidden Markov Model



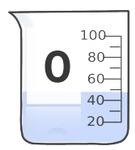
T = 0 1 2

P(0 2 20) = ?

Hidden Markov Model

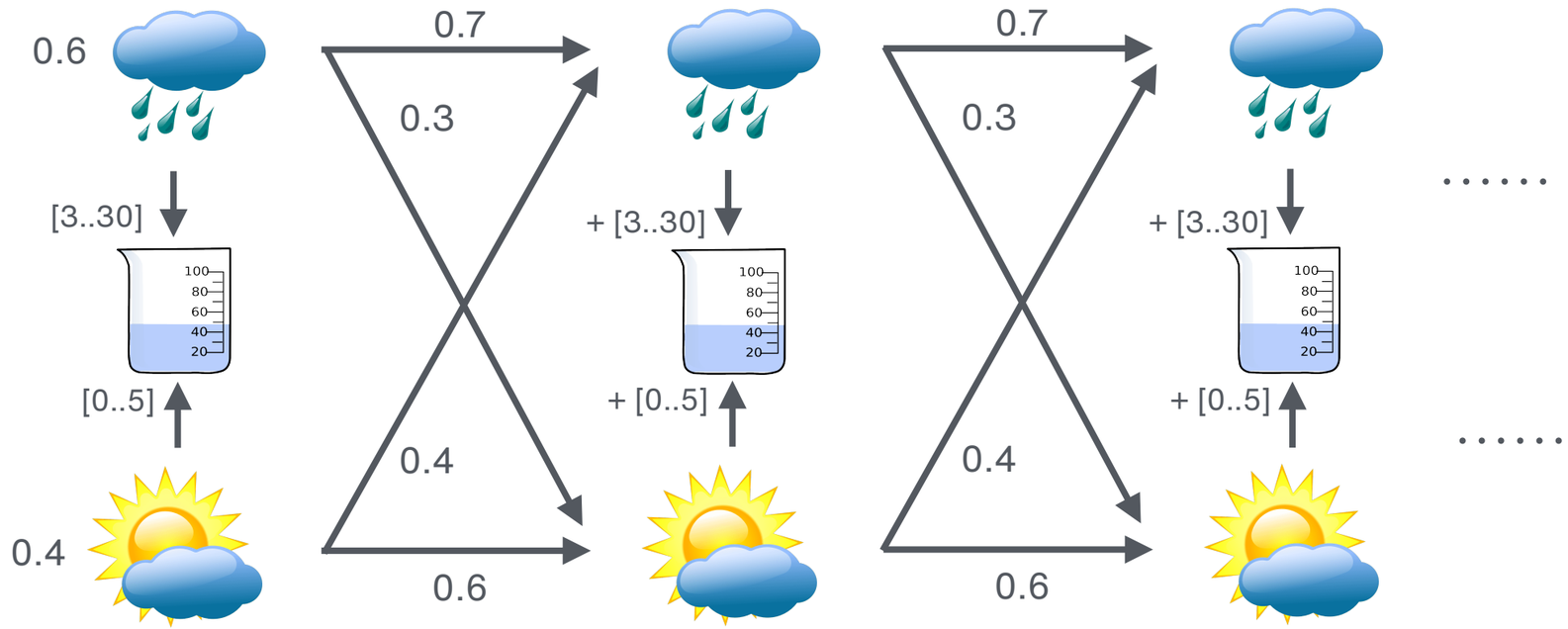


T = 0 1 2

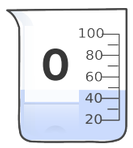
P(  ) = ?

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.

state ~ [[rainy, 0.7], [sunny, 0.3]] @ T+1 :-
    state=rainy @ T.

obs ~ [R+3..R+30] @ T :-
    state=rainy @ T, T > 0, obs=R @ T-1.

?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2.
```

Hidden Markov Model

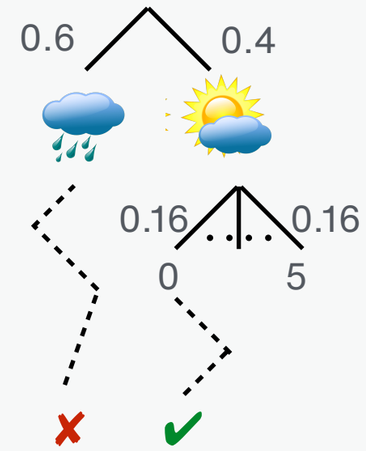


T = 0 1 2

P(  ) = ?

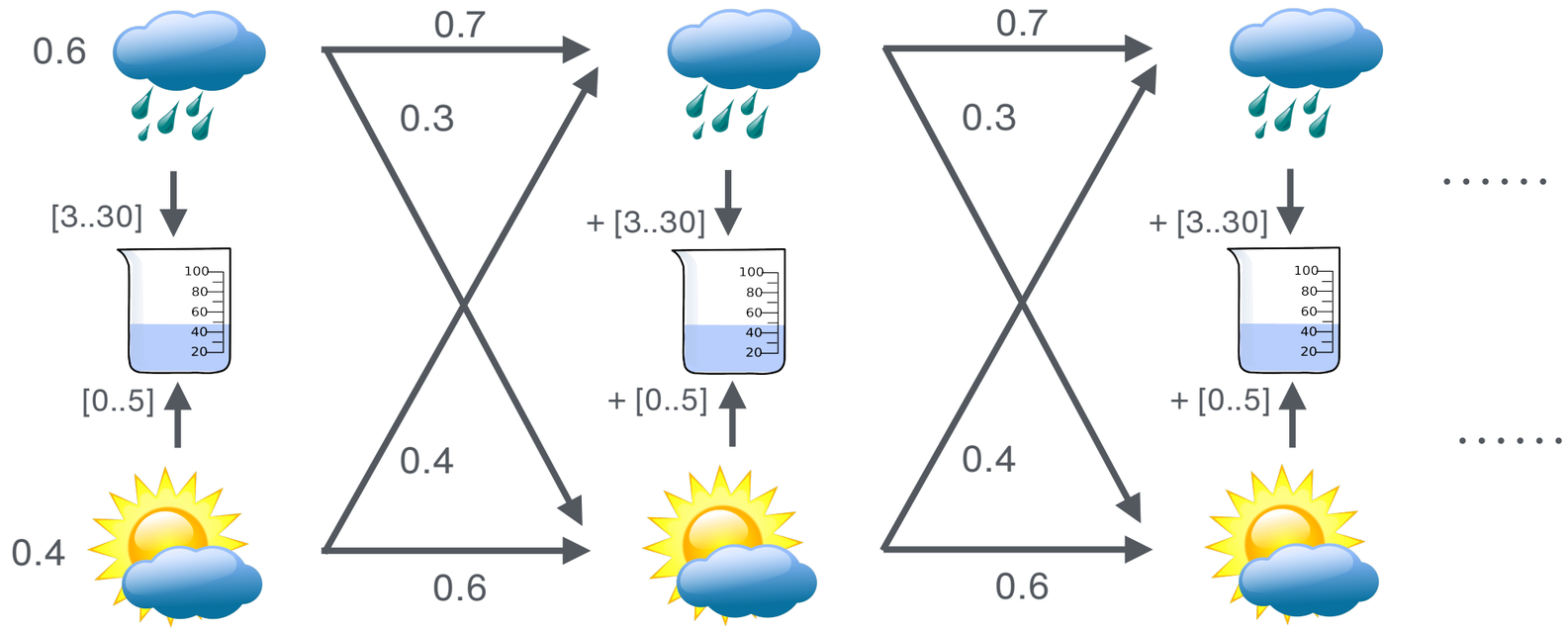
```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.

state ~ [[rainy, 0.7], [sunny, 0.3]] @ T+1 :-
    state=rainy @ T.

obs ~ [R+3..R+30] @ T :-
    state=rainy @ T, T > 0, obs=R @ T-1.

?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2.
```

Distribution Semantics

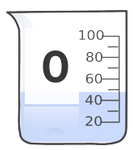


P(query) = $\sum_{✔}$ P(✔)

Hidden Markov Model

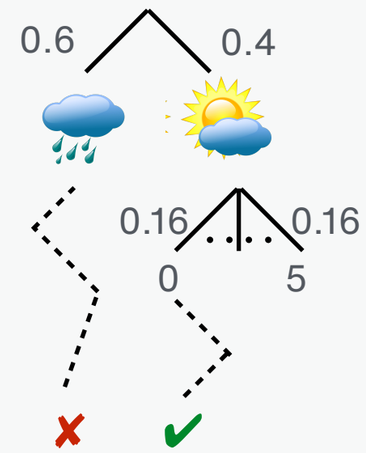


T = 0 1 2

P(  ) = 0.000119

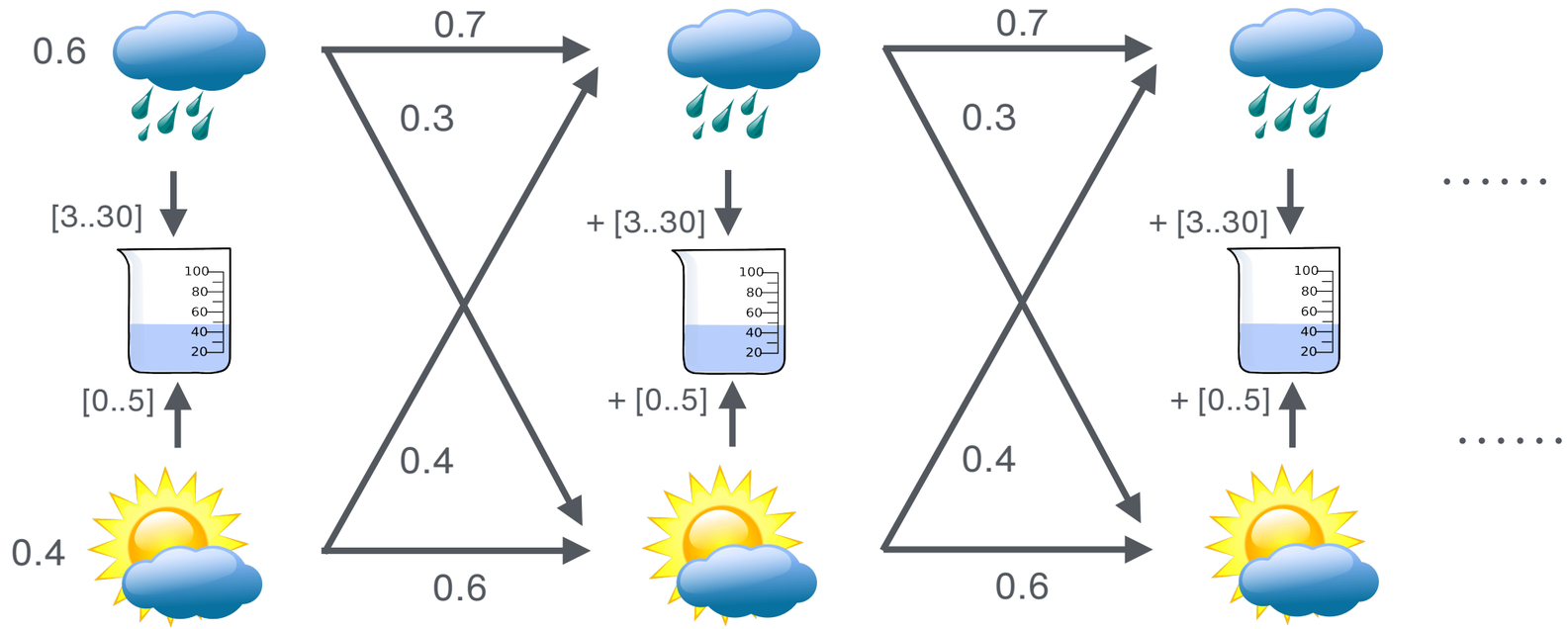
```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.

state ~ [[rainy, 0.7], [sunny, 0.3]] @ T+1 :-
    state=rainy @ T.

obs ~ [R+3..R+30] @ T :-
    state=rainy @ T, T > 0, obs=R @ T-1.

?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2.
```

Distribution Semantics

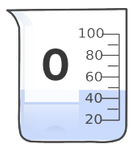


P(query) = $\sum_{✔}$ P(✔)

Hidden Markov Model

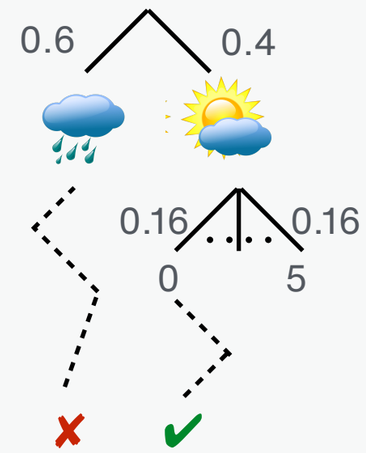


T = 0 1 2

P(  ) = 0.000119

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.

state ~ [[rainy, 0.7], [sunny, 0.3]] @ T+1 :-
    state=rainy @ T.

obs ~ [R+3..R+30] @ T :-
    state=rainy @ T, T > 0, obs=R @ T-1.

?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2.
```

Distribution Semantics



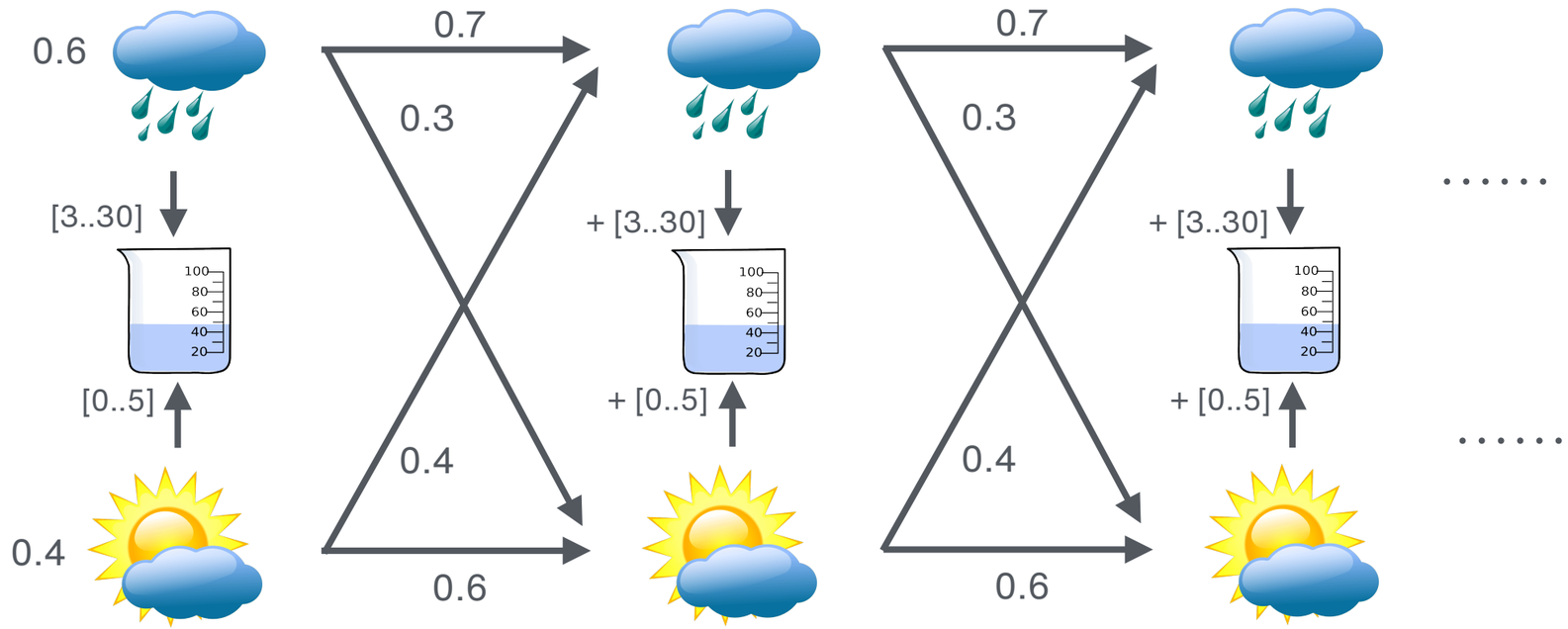
P(query) = $\sum_{✔}$ P(✔)

Computing query success probabilities

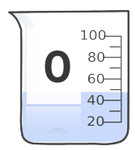
(1) Program grounding

(2) Variable elimination on ground program
(≈ SLD resolution + tabling)

Hidden Markov Model

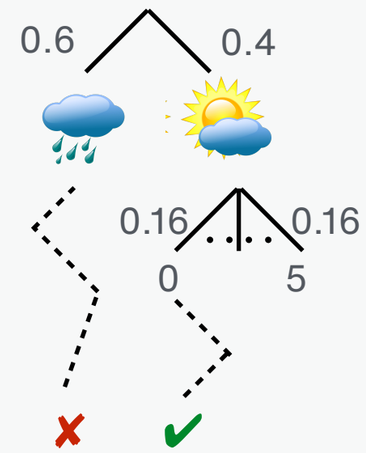


T = 0 1 2

P(  ) = 0.000119

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.

state ~ [[rainy, 0.7], [sunny, 0.3]] @ T+1 :-
    state=rainy @ T.

obs ~ [R+3..R+30] @ T :-
    state=rainy @ T, T > 0, obs=R @ T-1.

?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2.
```

Distribution Semantics



P(query) = $\sum_{✔}$ P(✔)

Computing query success probabilities

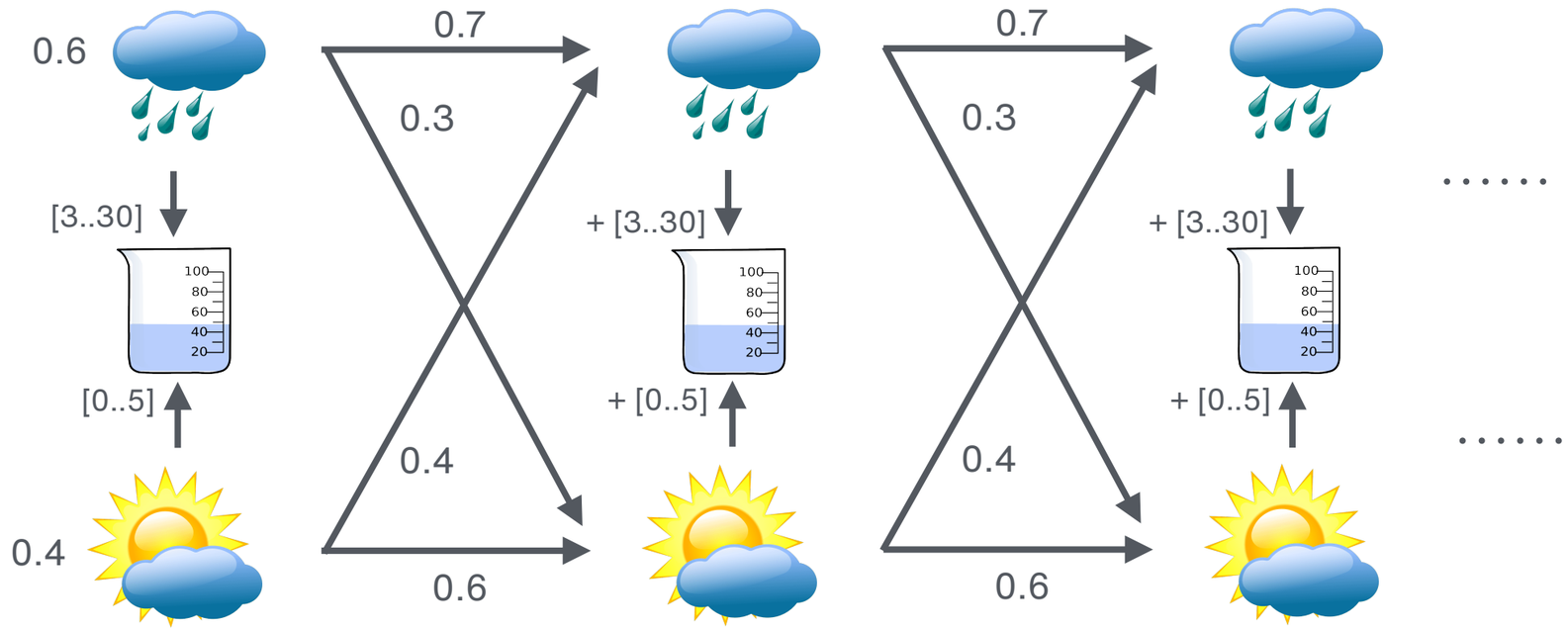
(1) Program grounding

(2) Variable elimination on ground program

(≈ SLD resolution + tabling)

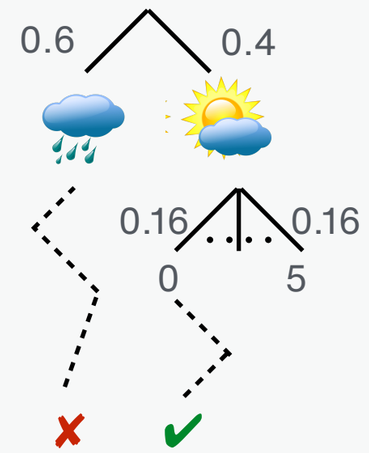
Make bodies h :- b1. h :- b2.[5] disjoint

Hidden Markov Model




```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.

state ~ [[rainy, 0.7], [sunny, 0.3]] @ T+1 :-
    state=rainy @ T.

obs ~ [R+3..R+30] @ T :-
    state=rainy @ T, T > 0, obs=R @ T-1.

?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2.
```

Distribution Semantics



P(query) = $\sum_{✔}$ P(✔)

Computing query success probabilities

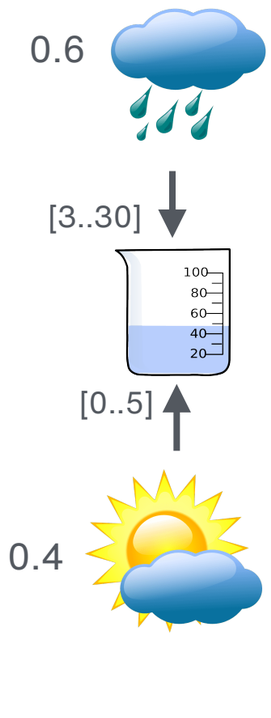
(1) Program grounding ← Next

(2) Variable elimination on ground program

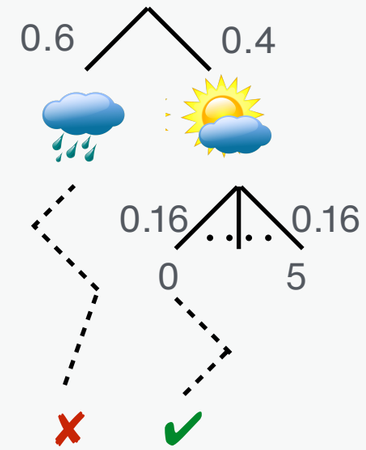
(≈ SLD resolution + tabling)

Make bodies h :- b1. h :- b2.[5] disjoint

Hidden Markov Model - Bottom-Up Grounding



Distribution Semantics



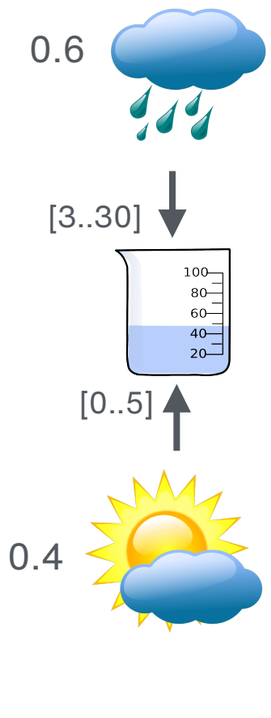
P(query) = Σ P(✔)

(Already grounded) program rules T = 0

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.
obs ~ [3..30] @ 0 :- state=rainy @ 0.
obs ~ [0..5]  @ 0 :- state=sunny @ 0.
```

```
?- obs=0 @ 0, obs=2 @ 1, obs=20 @ 2.
```

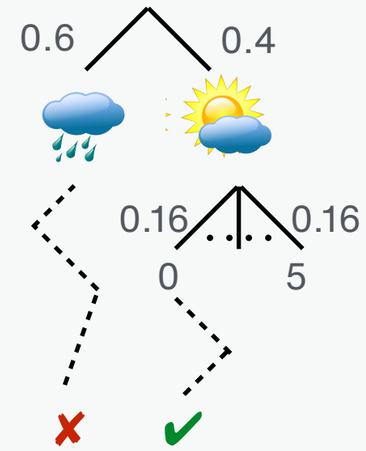
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



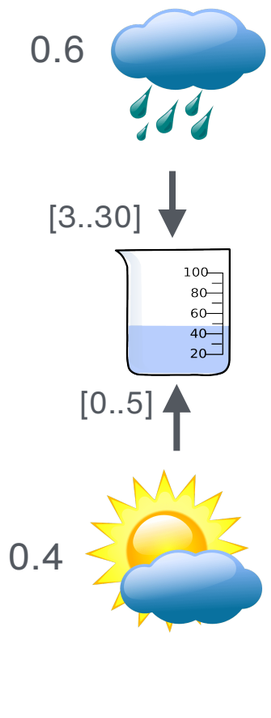
P(query) = Σ P(✔)

(Already grounded) program rules T = 0

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.
obs ~ [3..30] @ 0 :- state=rainy @ 0.
obs ~ [0..5]  @ 0 :- state=sunny @ 0.
```

```
?- obs=0 @ 0, obs=2 @ 1, obs=20 @ 2.
```

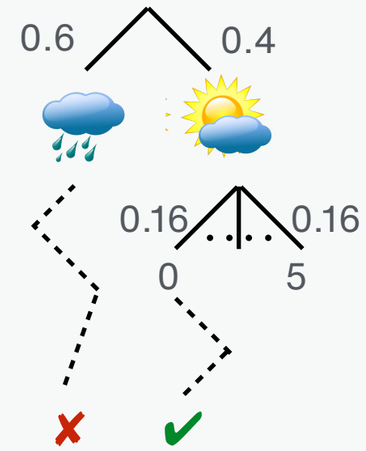
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

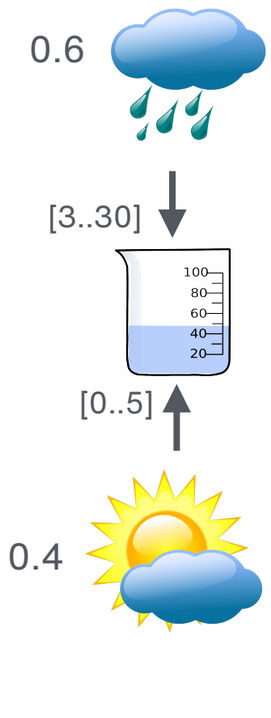
(Already grounded) program rules T = 0

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.
obs ~ [3..30] @ 0 :- state=rainy @ 0.
obs ~ [0..5]  @ 0 :- state=sunny @ 0.
```

Strengthen query by regression

```
?- obs=0 @ 0, obs=2 @ 1, obs=20 @ 2, state=sunny @ 0.
```

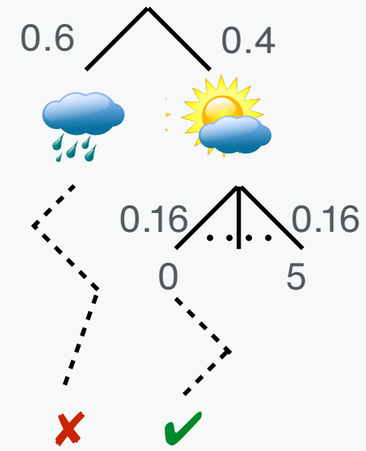
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

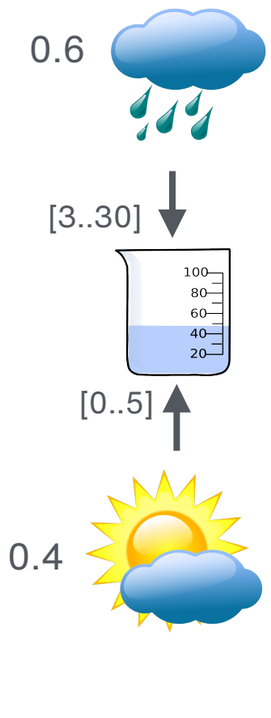
(Already grounded) program rules T = 0

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.
obs ~ [3..30] @ 0 :- state=rainy @ 0.
obs ~ [0..5]  @ 0 :- state=sunny @ 0.
```

Strengthen query by regression

```
?- obs=0 @ 0, obs=2 @ 1, obs=20 @ 2, state=sunny @ 0.
```

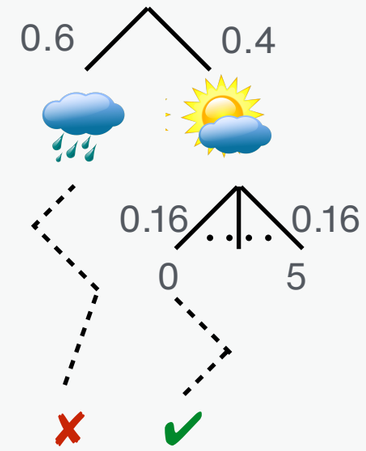
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

(Already grounded) program rules T = 0

state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.

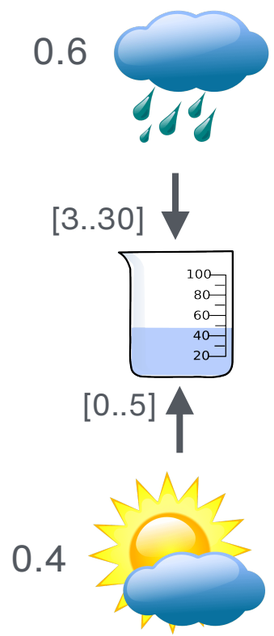
~~obs ~ [3..30] @ 0 :- state=rainy @ 0.~~ IP pruning

obs ~ [0..5] @ 0 :- state=sunny @ 0.

Strengthen query by regression

?- obs=0 @ 0, obs=2 @ 1, obs=20 @ 2, state=sunny @ 0.

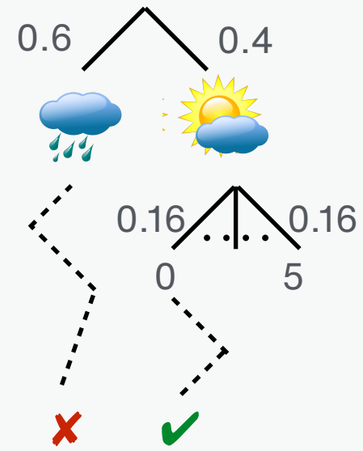
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

(Already grounded) program rules T = 0 → **Domain after T = 0**

```
state ~ [[rainy, 0.6], [sunny, 0.4]] @ 0.
obs ~ [3..30] @ 0 :- state=rainy @ 0.
obs ~ [0..5]  @ 0 :- state=sunny @ 0.
```

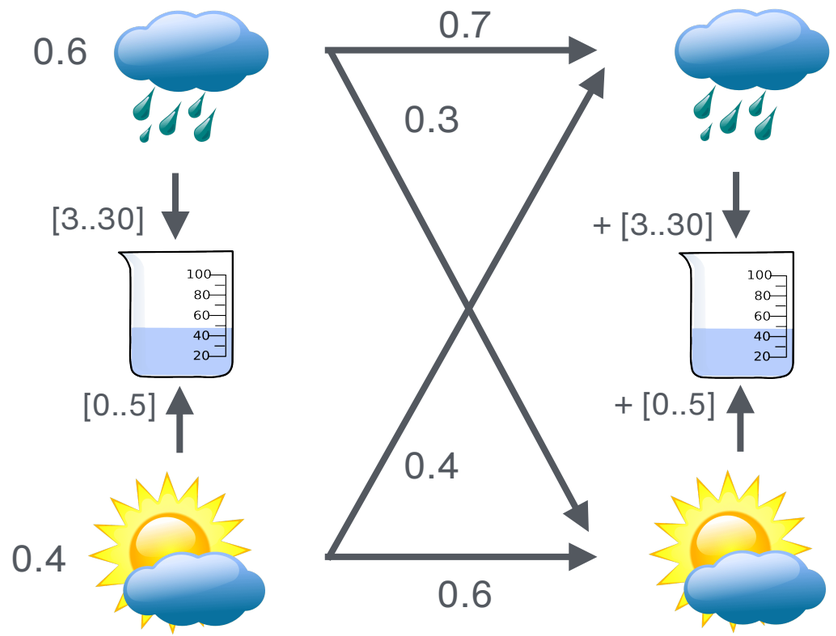
IP pruning

Strengthen query by regression

```
?- obs=0 @ 0, obs=2 @ 1, obs=20 @ 2, state=sunny @ 0.
```

```
state = rainy @ 0.
state = sunny @ 0.
obs = 0 @ 0.
obs = 1 @ 0.
:
obs = 5 @ 0.
```

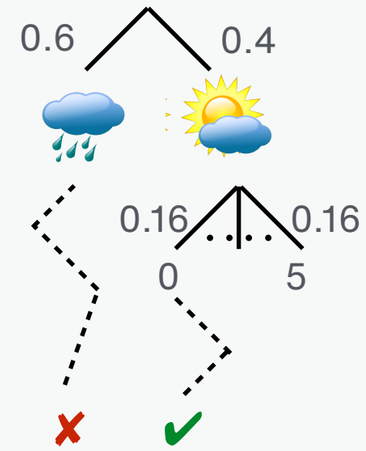
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with ⋃ heads

Distribution Semantics



P(query) = Σ P(✔)

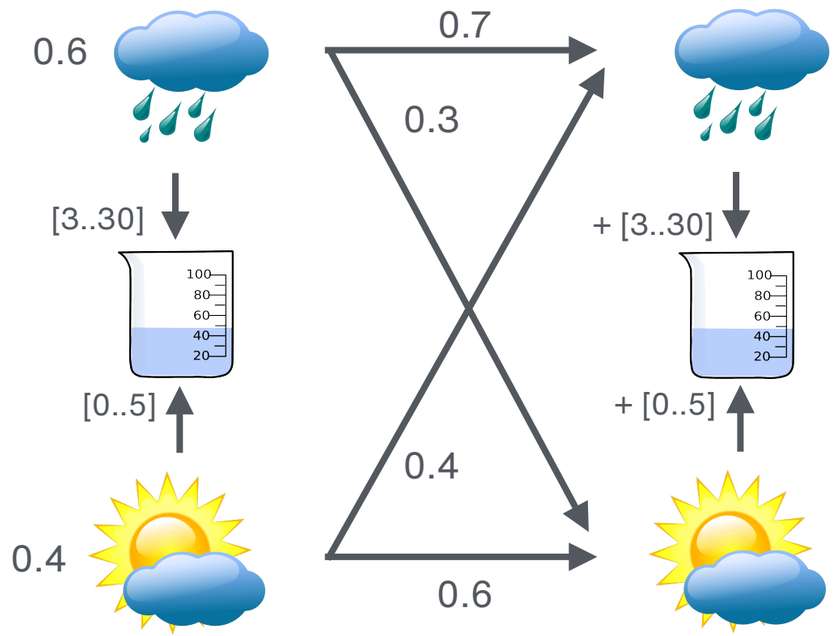
Domain T = 1

```
obs = 0 @ 0.
obs = 1 @ 0.
:
obs = 5 @ 0.
state = rainy @ 1.
state = sunny @ 1.
```

```
obs ~ [R+3..R+30] @ T :-
    state=rainy @ T,
    T > 0,
    obs=R @ T-1.
```

```
?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2, state=sunny @ 0.
```

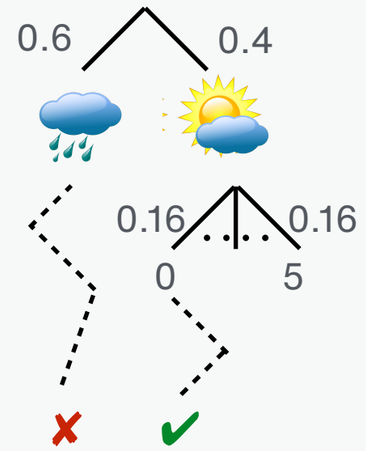
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

Domain T = 1 ⟶ **Grounded program rules T = 1**

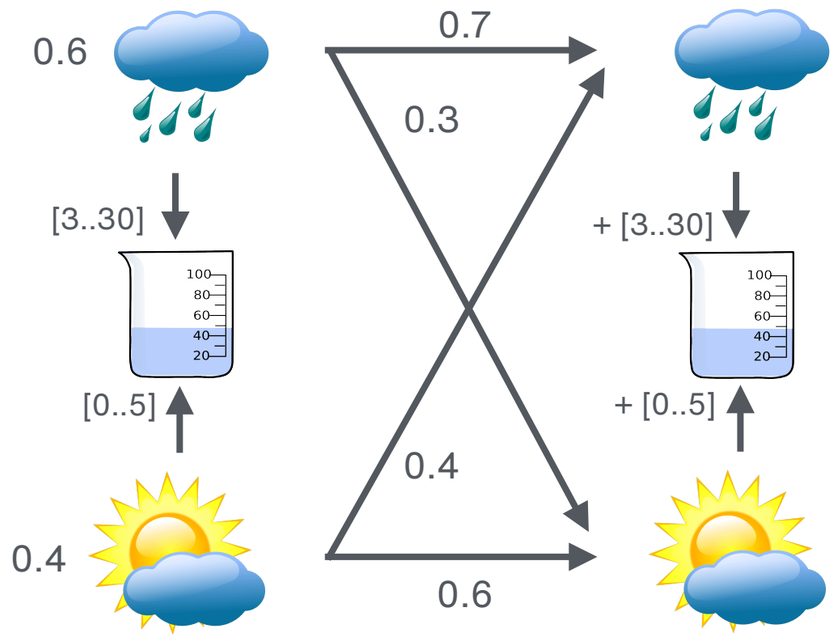
```
obs = 0 @ 0.
obs = 1 @ 0.
:
obs = 5 @ 0.
state = rainy @ 1.
state = sunny @ 1.
```

```
obs ~ [3..30]  @ 1 :- state=rainy @ 1, obs=0 @ 0.
obs ~ [4..31]  @ 1 :- state=rainy @ 1, obs=1 @ 0.
:
obs ~ [0..5]   @ 1 :- state=sunny @ 1, obs=0 @ 0.
obs ~ [1..6]   @ 1 :- state=sunny @ 1, obs=1 @ 0.
:
```

```
obs ~ [R+3..R+30] @ T :-
    state=rainy @ T,
    T > 0,
    obs=R @ T-1.
```

```
?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2, state=sunny @ 0.
```

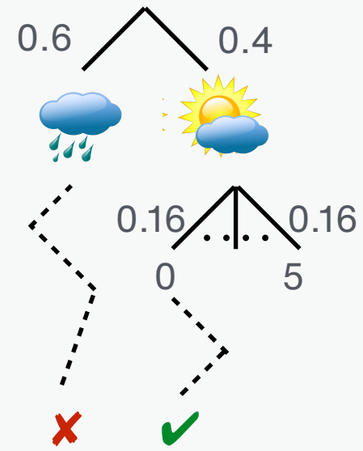
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

Domain T = 1 ⟶ **Grounded program rules T = 1**

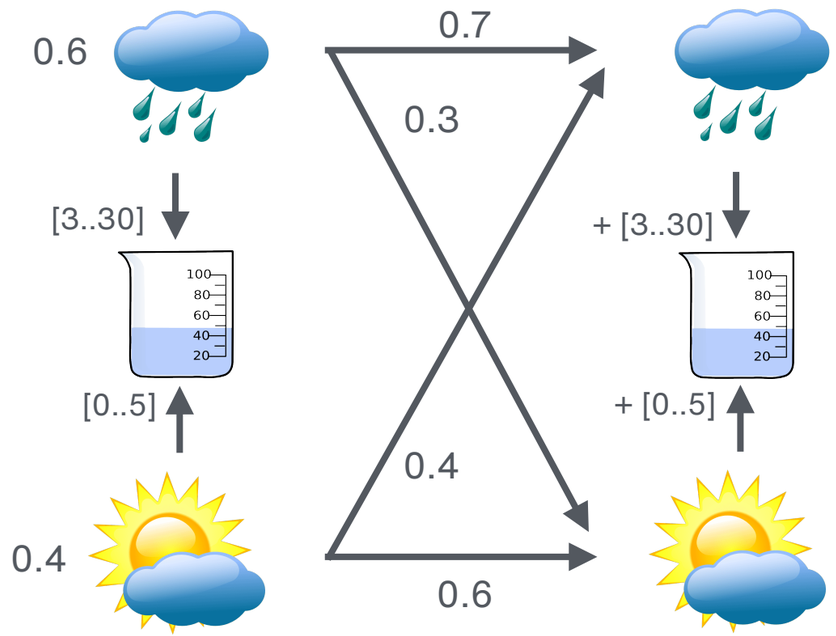
```
obs = 0 @ 0.
obs = 1 @ 0.
:
obs = 5 @ 0.
state = rainy @ 1.
state = sunny @ 1.
```

```
obs ~ [3..30]  @ 1 :- state=rainy @ 1, obs=0 @ 0.
obs ~ [4..31]  @ 1 :- state=rainy @ 1, obs=1 @ 0.
:
obs ~ [0..5]   @ 1 :- state=sunny @ 1, obs=0 @ 0.
obs ~ [1..6]   @ 1 :- state=sunny @ 1, obs=1 @ 0.
:
```

```
obs ~ [R+3..R+30] @ T :-
    state=rainy @ T,
    T > 0,
    obs=R @ T-1.
```

```
?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2, state=sunny @ 0.
```

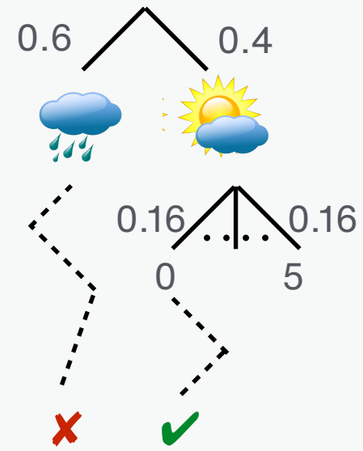
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

Domain T = 1 ⟶ **Grounded program rules T = 1**

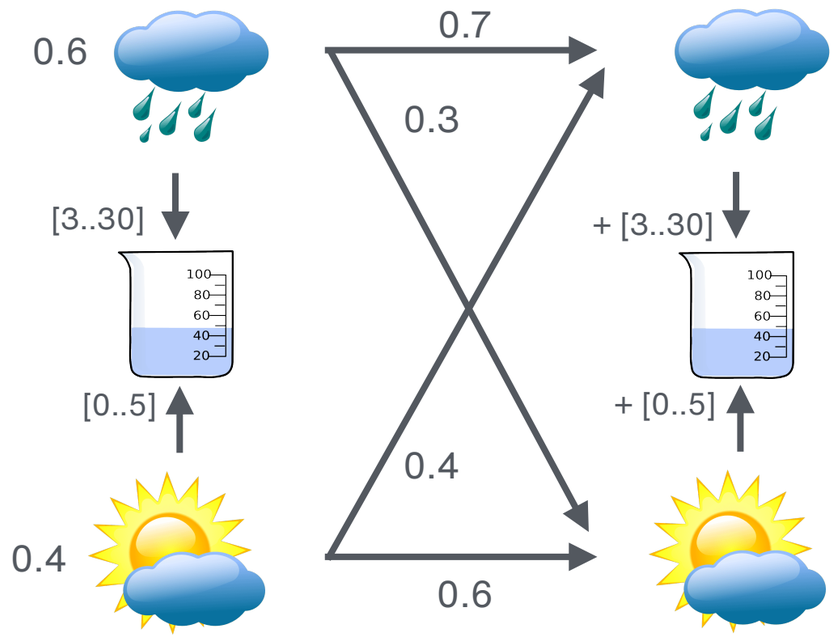
```
obs = 0 @ 0.
obs = 1 @ 0.
:
obs = 5 @ 0.
state = rainy @ 1.
state = sunny @ 1.
```

```
obs ~ [3..30]  @ 1 :- state=rainy @ 1, obs=0 @ 0.
obs ~ [4..31]  @ 1 :- state=rainy @ 1, obs=1 @ 0.
:
obs ~ [0..5]   @ 1 :- state=sunny @ 1, obs=0 @ 0.
obs ~ [1..6]   @ 1 :- state=sunny @ 1, obs=1 @ 0.
:
```

```
obs ~ [R+3..R+30] @ T :-
    state=rainy @ T,
    T > 0,
    obs=R @ T-1.
```

```
?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2, state=sunny @ 0.
```

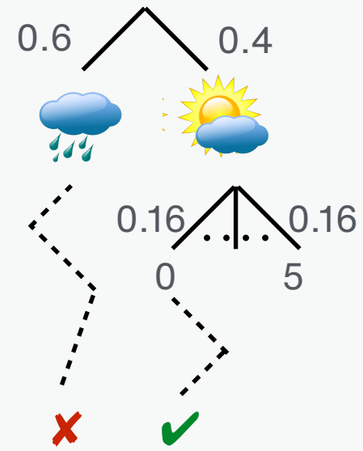
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

Domain T = 1 → **Grounded program rules T = 1**

```
obs = 0 @ 0.
obs = 1 @ 0.
:
obs = 5 @ 0.
state = rainy @ 1.
state = sunny @ 1.
```

```
obs ~ [3..30]  @ 1 :- state=rainy @ 1, obs=0 @ 0.
obs ~ [4..31]  @ 1 :- state=rainy @ 1, obs=1 @ 0.
:
obs ~ [0..5]   @ 1 :- state=sunny @ 1, obs=0 @ 0.
obs ~ [1..6]   @ 1 :- state=sunny @ 1, obs=1 @ 0.
:
```

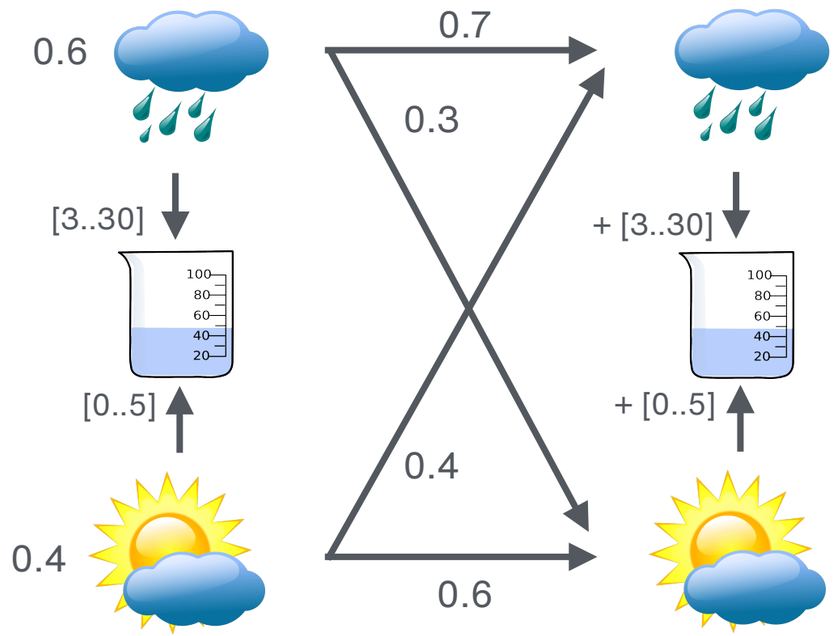
IP

IP

```
obs ~ [R+3..R+30] @ T :-
    state=rainy @ T,
    T > 0,
    obs=R @ T-1.
```

```
?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2, state=sunny @ 0.
```

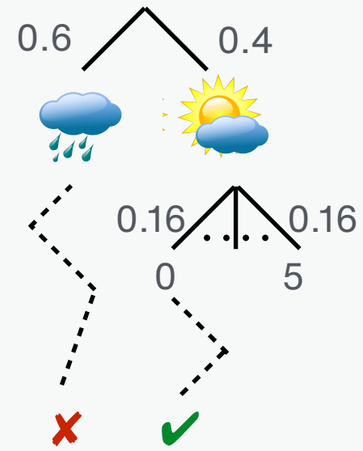
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with ⋃ heads

Distribution Semantics



P(query) = Σ P(✔)

Domain T = 1 ⟶ **Grounded program rules T = 1**

```
obs = 0 @ 0.
obs = 1 @ 0.
:
obs = 5 @ 0.
state = rainy @ 1.
state = sunny @ 1.
```

```
obs ~ [3..30]  @ 1 :- state=rainy @ 1, obs=0 @ 0.
obs ~ [4..31]  @ 1 :- state=rainy @ 1, obs=1 @ 0.
.
obs ~ [0..5]   @ 1 :- state=sunny @ 1, obs=0 @ 0.
obs ~ [1..6]   @ 1 :- state=sunny @ 1, obs=1 @ 0.
.
```

IP

IP

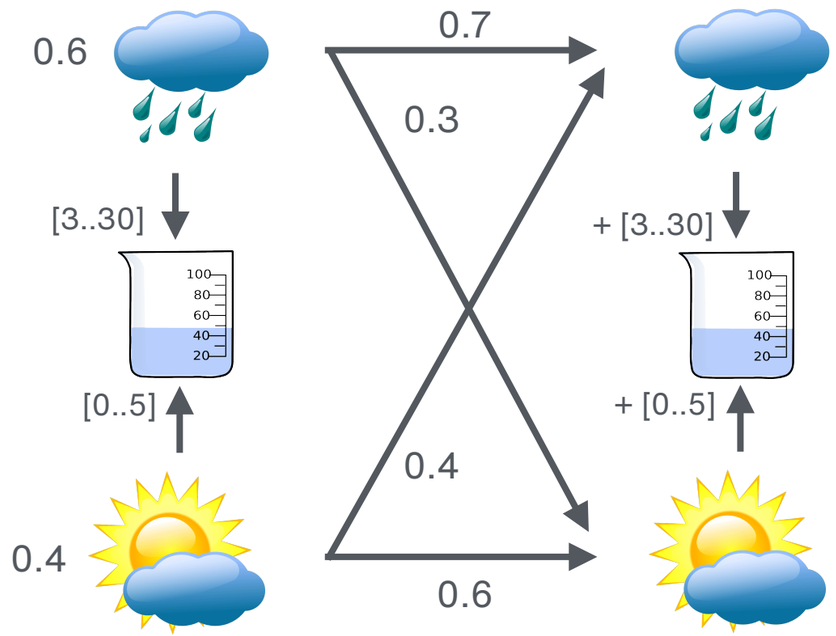
```
obs ~ [R+3..R+30] @ T :-
    state=rainy @ T,
    T > 0,
    obs=R @ T−1.
```

Inconsistency pruning: 62 -> 2 rules

(I.p. also applies to literals a and \+ a)

```
?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2, state=sunny @ 0.
```

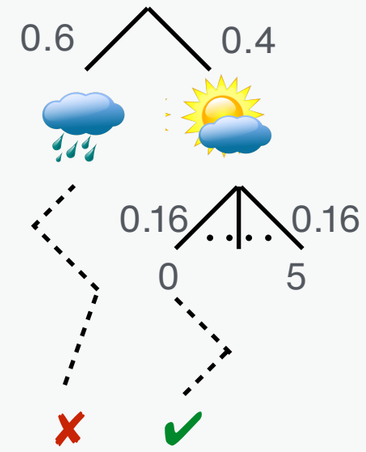
Hidden Markov Model - Bottom-Up Grounding



In increasing stratification order:

- Ground out program over current domain
- Query regression, inconsistency pruning
- Extend current domain with $\bigcup$ heads

Distribution Semantics



P(query) = Σ P(✔)

Domain T = 1 → **Grounded program rules T = 1**

```
obs = 0 @ 0.                obs ~ [3..30]  @ 1 :- state=rainy @ 1, obs=0 @ 0.
obs = 1 @ 0.                obs ~ [4..31]  @ 1 :- state=rainy @ 1, obs=1 @ 0.
:                           IP
obs = 5 @ 0.                :
state = rainy @ 1.          obs ~ [0..5]   @ 1 :- state=sunny @ 1, obs=0 @ 0.
state = sunny @ 1.          obs ~ [1..6]   @ 1 :- state=sunny @ 1, obs=1 @ 0.
                            IP
                            :
```

```
obs ~ [R+3..R+30] @ T :-
    state=rainy @ T,
    T > 0,
    obs=R @ T-1.
```

Inconsistency pruning: 62 -> 2 rules

(I.p. also applies to literals a and \+ a)

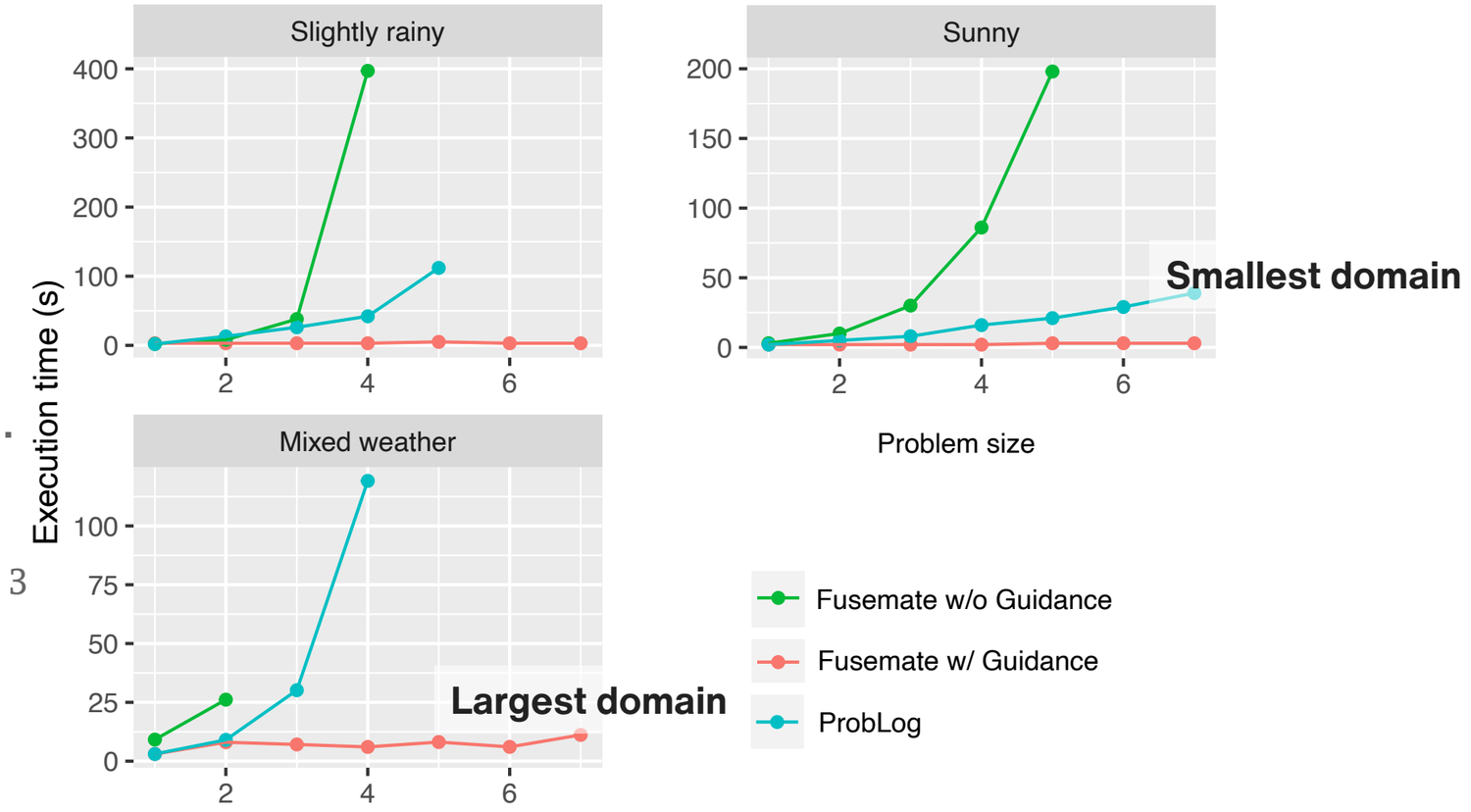
```
?- obs=0 @ 0, obs=4 @ 1, obs=20 @ 2, state=sunny @ 0.
```

Experimental Evaluation 1 - Hidden Markov Model

Runtime Results Fusemate vs ProbLog

Rainy/sunny example from above

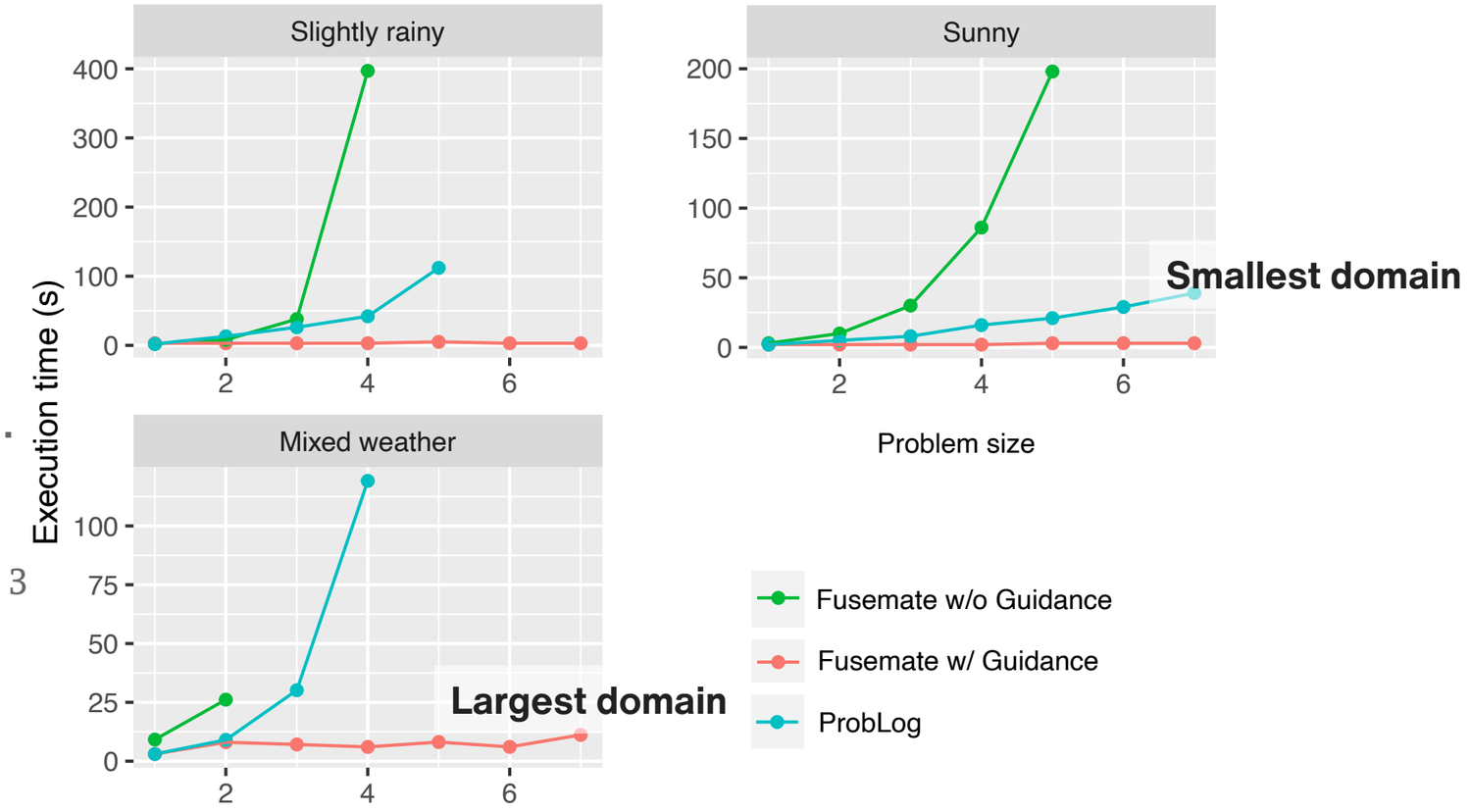
```
%% Queries for N=3
%% Sunny
?-state=X @ 3 | obs=0 @ 1, obs=0 @ 2, obs=0 @ 3.

%% Rainy
?-state=X @ 3 | obs=4 @ 1, obs=8 @ 2, obs=12 @ 3

%% Mixed
state=X @ 3 | obs=0 @ 1, obs=4 @ 2, obs=24 @ 3.
```



Experimental Evaluation 1 - Hidden Markov Model

Runtime Results Fusemate vs ProbLog

```
Rainy/sunny example from above

%% Queries for N=3
%% Sunny
?-state=X @ 3 | obs=0 @ 1, obs=0 @ 2, obs=0 @ 3.

%% Rainy
?-state=X @ 3 | obs=4 @ 1, obs=8 @ 2, obs=12 @ 3

%% Mixed
state=X @ 3 | obs=0 @ 1, obs=4 @ 2, obs=24 @ 3.
```



Grounding vs Inference - Mixed Weather

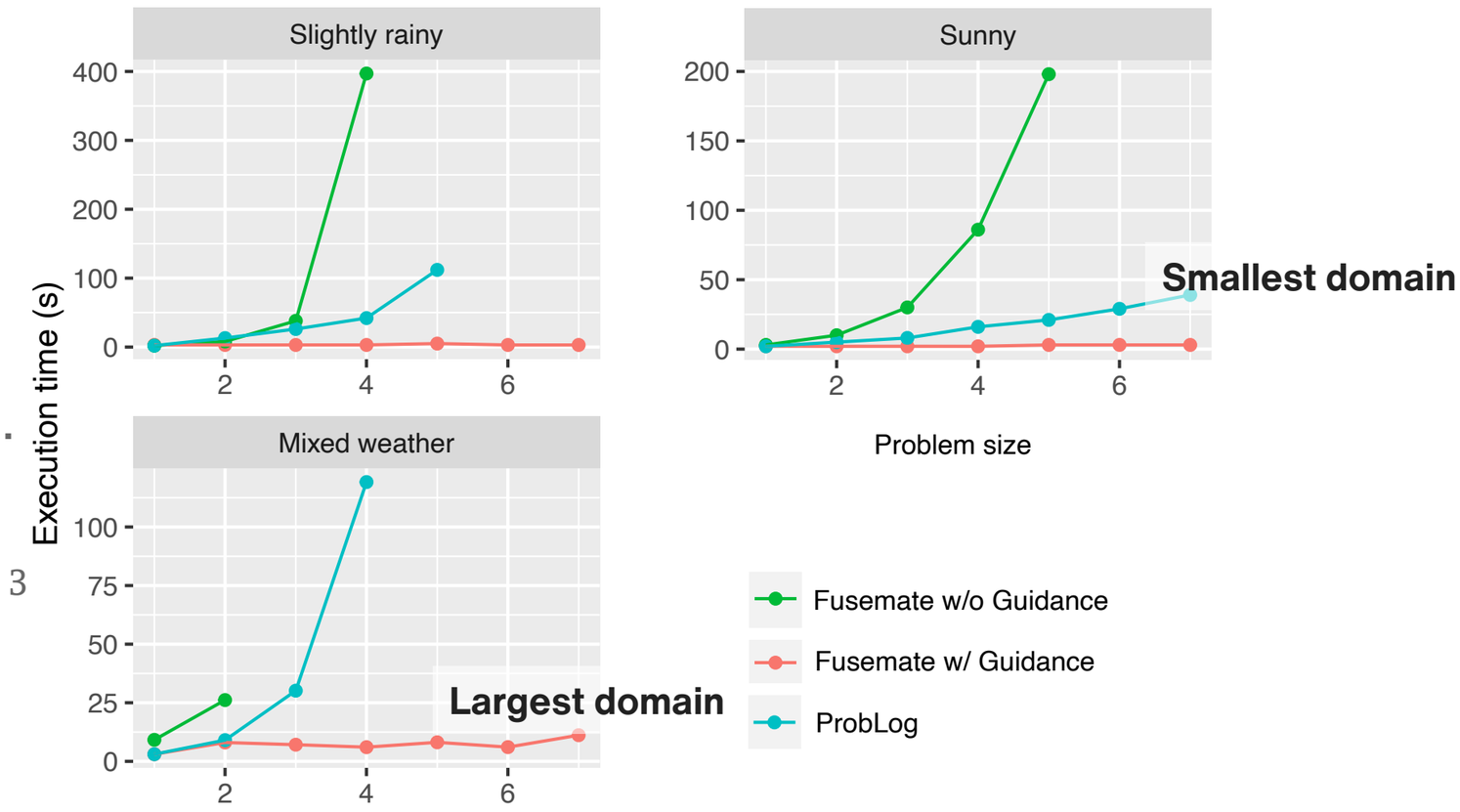
	Fusemate #ground rules		ProbLog		
N	query-guided	unguided	total time	grounding time	#ground rules
2	2200	6500	9.0	8.3	53
3	2270	12900	30	19	276
4	2300	21400	119	33	499
5	2400	32000		50	682
6	2470	45000		65	839
7	2500	60000		95	1068

Experimental Evaluation 1 - Hidden Markov Model

Runtime Results Fusemate vs ProbLog

Rainy/sunny example from above

```
%% Queries for N=3
%% Sunny
?-state=X @ 3 | obs=0 @ 1, obs=0 @ 2, obs=0 @ 3.

%% Rainy
?-state=X @ 3 | obs=4 @ 1, obs=8 @ 2, obs=12 @ 3

%% Mixed
state=X @ 3 | obs=0 @ 1, obs=4 @ 2, obs=24 @ 3.
```



Grounding vs Inference - Mixed Weather

	Fusemate #ground rules		ProbLog		
N	query-guided	unguided	total time	grounding time	#ground rules
2	2200	6500	9.0	8.3	53
3	2270	12900	30	19	276
4	2300	21400	119	33	499
5	2400	32000		50	682
6	2470	45000		65	839
7	2500	60000		95	1068

Fusemate:

Improved grounding pays off

ProbLog:

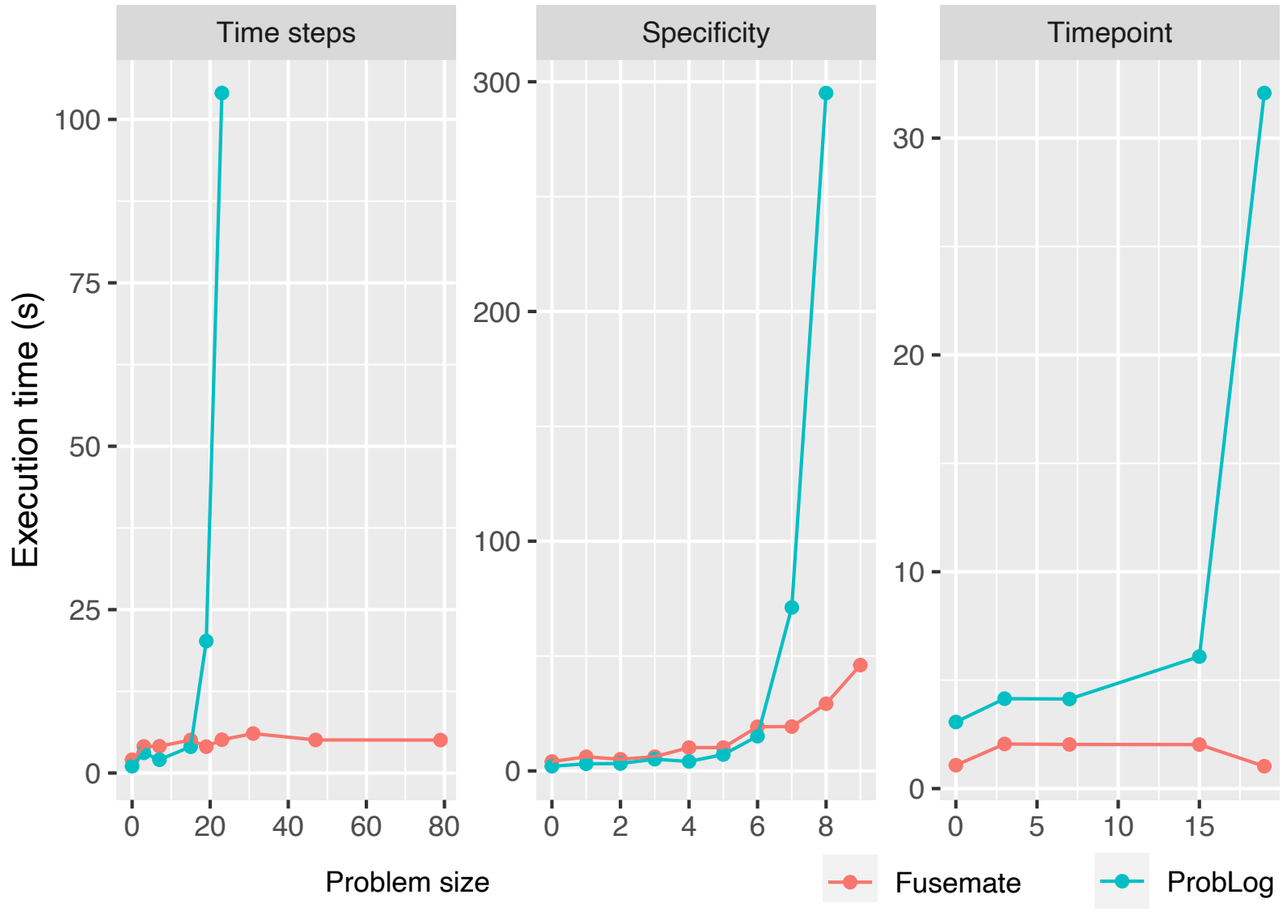
Grounding OK?

Bottleneck inference component?

Experimental Evaluation 2 - Markov Model

Runtime Results Fusemate vs ProbLog

```
%% Markov Model
in ~ [a, b, c] @ 0.
in ~ [[a, 0.9],[b, 0.05],
      [c, 0.05]] @ T+1 :- in=a @ T.
in ~ [[a, 0.7],[c, 0.3]] @ T+1 :- in=b @ T.
in ~ [[a, 0.8],[c, 0.2]] @ T+1 :- in=c @ T.

%% Time steps N = 20
?- in=a@0, in=a@1,.., in=a@20.

%% Specificity, N = 7
?- in=a@0, in=a@1, in=L2@2,..,in=L8 @ 8.

%% Timepoint, N = 20
?- in=a@23.
```



(ProbLog code from ProbLog tutorial web page)

Conclusions

Key idea

- Fix semantics of ‘=‘ as a right-unique relation
- Basis for pruning based on inconsistencies: x=5 and x=6 cannot be simultaneous true
- However assumption: all models of the program are consistent

Inconsistency pruning vs magic sets

- Magic sets: generate rules that can potentially reach the query
- Inconsistency pruning: prune rules that can impossibly reach the query
- Can be combined - future work

Extension: Inconsistency based pruning during inference

- Prune inconsistent queries as soon as derived by regression ?– … x=5, x=6, …
- Can improve performance considerable for less constrained queries ?– state=S @ 3 | obs = 20 @ 3.
- See paper for details

Implementation

- In Scala; two-way integration; see paper for download URL

Future work

- More comparison with ProbLog; swap grounding and inference components
- Positive cycles