Frame Problem

- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0))$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s)$$
$$\rightarrow At(o,x,Result(Go(y,z),s)).$$

This says moving does not affect the gold when it is not held. Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).

Frame Problem

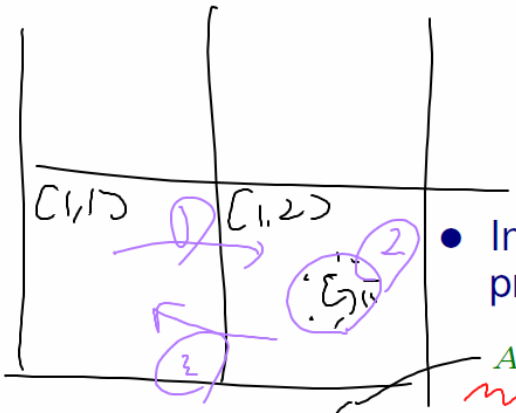
- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0))$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s)$$
$$\rightarrow At(o,x,Result(Go(y,z),s)).$$

This says moving does not affect the gold when it is not held. Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).



Frame Problem

- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0))$$

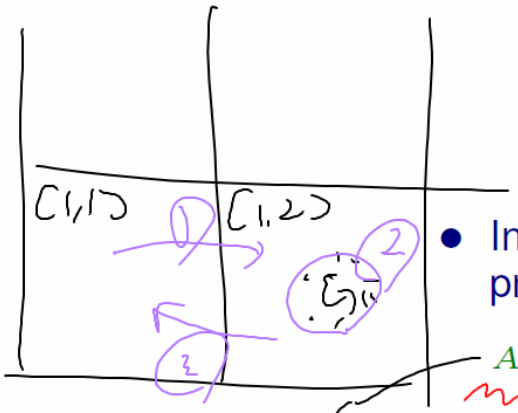
- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o, x, s) \wedge (o \neq Agent) \wedge \neg Holding(o, s)$$
$$\rightarrow At(o, x, Result(Go(y, z), s)).$$

This says moving does not affect the gold when it is not held.

Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).

To ensure that you have to actually add more description of these effects of moving around this stuff that you're holding



Frame Problem

- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

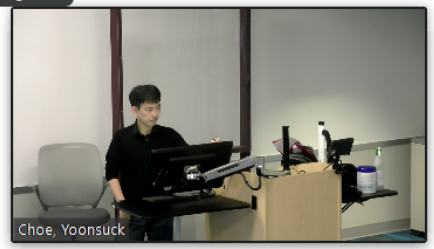
$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0))$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s)$$
$$\rightarrow At(o,x,Result(Go(y,z),s)).$$

This says moving does not affect the gold when it is not held.

Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).



Frame Problem

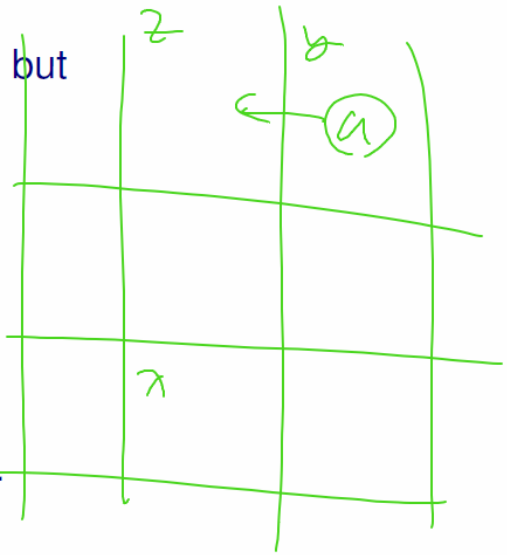
- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0)$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s)$$
$$\rightarrow At(o,x,Result(Go(y,z),s)).$$

This says moving does not affect the gold when it is not held. Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).



So this is called the frame axiom. So what other consequen

Frame Problem

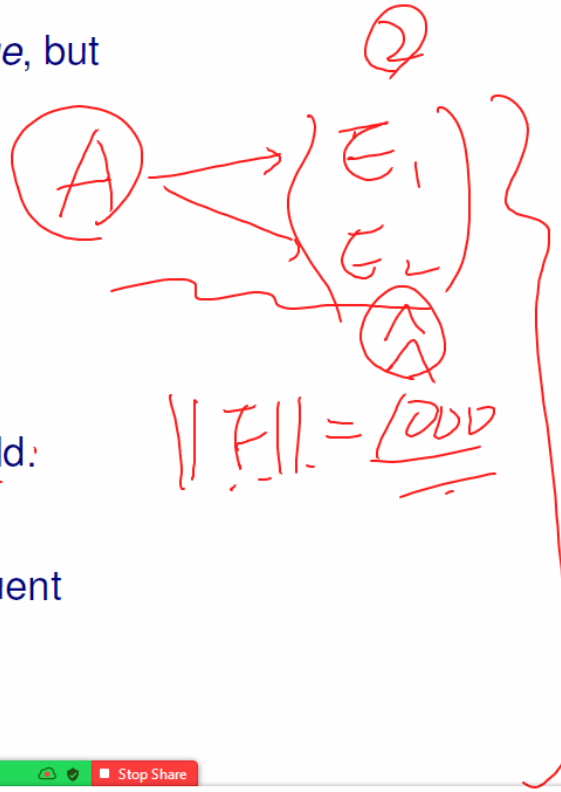
- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1, 1], Result([Go([1, 1], [1, 2]), Grab(G_1), Go([1, 2], [1, 1])], S_0)$$

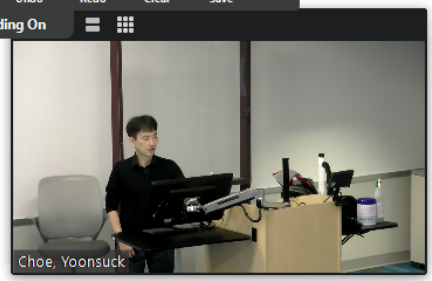
- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o, x, s) \wedge (o \neq Agent) \wedge \neg Holding(o, s) \rightarrow At(o, x, Result(Go(y, z), s)).$$

This says moving does not affect the gold when it is not held: Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).

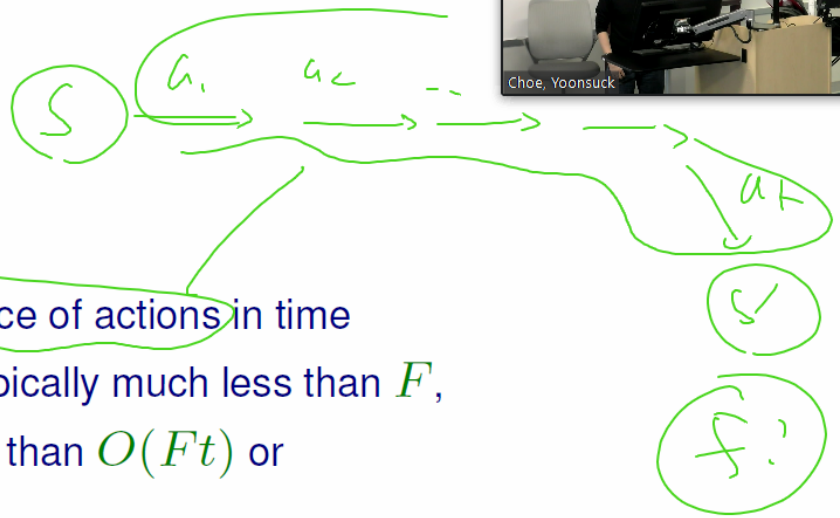


And the number of pretty kids. So if you have to actually have frame actions for the number of actions times this, then it gets pretty anything



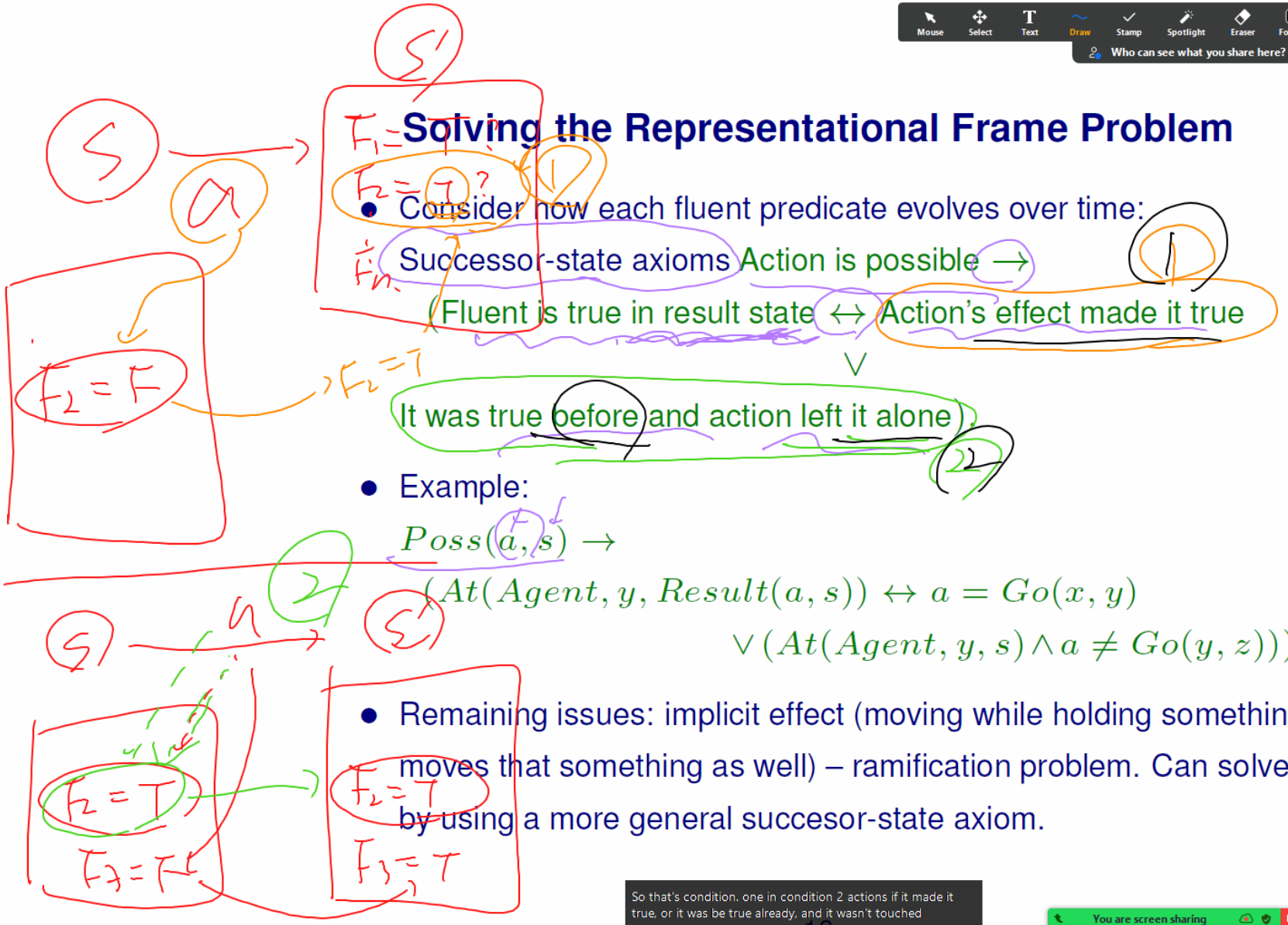
Two Frame Problems

- Representational frame problem:
  Explained in the previous slide
- Inferential frame problem:
  Projection of results of a t-step sequence of actions in time $O(Et)$ (E is the number of effects, typically much less than F, the number of fluent predicates), rather than $O(Ft)$ or $O(AEt)$.



Two Frame Problems

- Representational frame problem:
  Explained in the previous slide
- Inferential frame problem:
  Projection of results of a t-step sequence of actions in time $O(Et)$ (E is the number of effects, typically much less than F, the number of fluent predicates), rather than $O(Ft)$ or $O(AEt)$.



Solving the Representational Frame Problem

- Consider how each fluent predicate evolves over time:

  Successor-state axioms Action is possible $\rightarrow$

  (Fluent is true in result state $\leftrightarrow$ Action's effect made it true

  $\vee$

  It was true before and action left it alone).

- Example:

  $Poss(a, s) \rightarrow$

  $(At(Agent, y, Result(a, s)) \leftrightarrow a = Go(x, y)$

  $\vee (At(Agent, y, s) \wedge a \neq Go(y, z))).$

- Remaining issues: implicit effect (moving while holding something moves that something as well) – ramification problem. Can solve by using a more general succesor-state axiom.

Solving the Representational Frame Problem

- Consider how each fluent predicate evolves over time:

  Successor-state axioms Action is possible $\rightarrow$

  (Fluent is true in result state $\leftrightarrow$ Action's effect made it true

  $\vee$

  It was true before and action left it alone).

- Example:

  $Poss(a, s) \rightarrow$

  $(At(Agent, y, Result(a, s)) \leftrightarrow a = Go(x, y)$

  $\vee (At(Agent, y, s) \wedge a \neq Go(y, z))).$

- Remaining issues: implicit effect (moving while holding something moves that something as well) – ramification problem. Can solve by using a more general succesor-state axiom.

1.2 Substitution and Unification

Problem 4 (9 pts): Apply the following substitutions to the expressions (3 point

1. Apply $\{x/f(A)\}$ to $P(x, y) \vee Q(x)$.
2. Apply $\{x/A, y/f(z)\}$ to $P(x, y) \vee Q(x)$.
3. Apply $\{x/f(y), y/B\}$ to $P(x, y) \vee Q(x)$

Problem 5 (12 pts): For each of the following, example, given $P(A)$ and $P(x)$, the unifier wou expressions is not unifiable, indicate so. (3 poi set D_k for each iteration, and the σ_k.

1. $P(x, f(B))$ and $P(A, f(y))$
2. $P(y, y)$ and $P(x, f(A))$
3. $P(f(y), x, y)$ and $P(f(g(w)), w, g(A))$
4. $P(A, f(y), y, A)$ and $P(x, f(g(x)), g(B)$

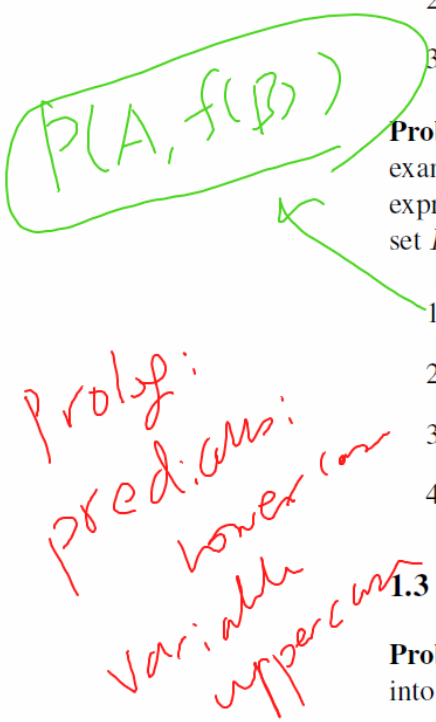
1.3 Proof by resolution

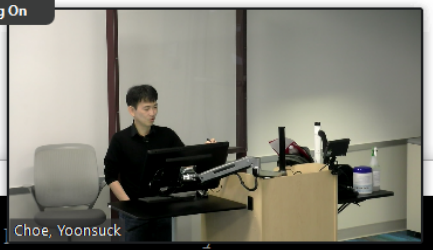
Problem 6 (16 pts): (1) Translate the follow into a canonical form, and (3) prove the theoren

Given:

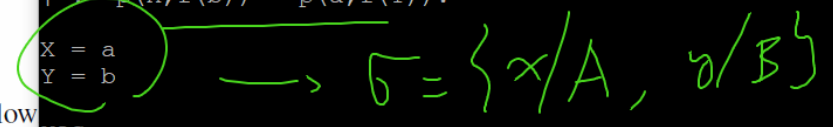
1. If a child is young, then he/she is adored.
2. Every child who is adored and fed well is happy.
3. Every child is fed well.

Show that the following is a logical consequence of the above:



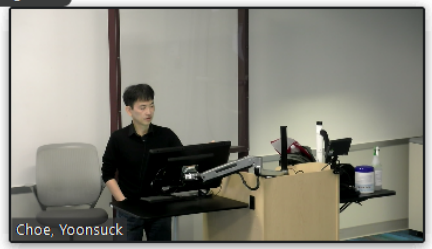



```
linux2.cse.tamu.edu - PuTTY
|
| If you get the error message: Failed to create
| Then you will need to first create your home directory. Directions are avail
> ble in your browser by visiting the following web page: http://bit.ly/linuxh
> medoc
|
| COMPUTER NAME: linux2.cs.tamu.edu
|
  End of banner message from server
  choe@sun.cs.tamu.edu's password:
linux2:/home/faculty/c/choe>
linux2:/home/faculty/c/choe>
linux2:/home/faculty/c/choe> gprolog
GNU Prolog 1.4.4 (64 bits)
Compiled Jan 15 2014, 20:38:19 with gcc
By Daniel Diaz
Copyright (C) 1999-2013 Daniel Diaz
| ?- p(X,f(b)) = p(a,f(Y)).

X = a
Y = b

yes
| ?-
```



You get p. of A. F. of B. a single expression. So you got the unified expression, but it only shows you she's just the answer. So you have to

You are screen sharing Stop Share



Solving the Representational Frame Problem

- Consider how each fluent predicate evolves over time:

  Successor-state axioms Action is possible $\rightarrow$

  (Fluent is true in result state $\leftrightarrow$ Action's effect made it true

  $\vee$

  It was true before and action left it alone).

- Example:

$$Poss(a, s) \rightarrow$$
$$(At(Agent, y, Result(a, s)) \leftrightarrow a = Go(x, y)$$
$$\vee (At(Agent, y, s) \wedge a \neq Go(y, z))).$$

- Remaining issues: implicit effect (moving while holding something moves that something as well) – ramification problem. Can solve by using a more general succesor-state axiom.

Solving the Representational Frame Problem

- Consider how each fluent predicate evolves over time:

  Successor-state axioms Action is possible $\rightarrow$

  (Fluent is true in result state $\leftrightarrow$ Action's effect made it true

  $\vee$

  It was true before and action left it alone).

- Example:

$$Poss(a, s) \rightarrow$$
$$(At(Agent, y, Result(a, s)) \leftrightarrow a = Go(x, y)$$
$$\vee (At(Agent, y, s) \wedge a \neq Go(y, z))).$$

- Remaining issues: implicit effect (moving while holding something moves that something as well) – ramification problem. Can solve by using a more general succesor-state axiom.

Okay, school. Still, there are remaining issues, even if you have these things laid out their implicit effects like moving while holding on something

Solving the Inferential Frame Problem

- Typical frame axiom: $Poss(a, s) \rightarrow$
  $F_i(Result(a, s)) \leftrightarrow (a = A_1 \vee a = A_2 ...)$
  $\vee (F_i(s) \wedge (a \neq A_3) \wedge (a \neq A_4) ...)$

- Several actions that make the fluent true and several that make the fluent false: Formalize using the predicate $PosEffect(a, F_i)$ and $NegEffect(a, F_i)$.
  $Poss(a, s) \rightarrow$
  $F_i(Result(a, s)) \leftrightarrow PosEffect(a, F_i)$
  $\vee [F_i(s) \wedge \neg NegEffect(a, F_i)]$
  $PosEffect(A_1, F_i), PosEffect(A_1, F_i)$
  $NegEffect(A_3, F_i), NegEffect(A_4, F_i)$
  * This can be done efficiently: get current action, and fetch its effects, then update those fluents $O(Et)$.

Solving the Inferential Frame Problem

- Given a t-step plan p ($S_t = Result(p, S_0)$), decide which fluents are true in S_t.
- We need to consider each of the F frame axiom of each time step t.
- Axioms have an average size of AE/F, we have an $O(AEt)$ inferential work. Most of the work is done copying unchanged fluents from time step to time step.
- Solutions: use fluent calculus rather than situation calculus, or make the process more efficient.

So we'll skip that

Solving the Inferential Frame Problem

- Typical frame axiom: $Poss(a, s) \rightarrow$

$$F_i(Result(a, s)) \leftrightarrow (a = A_1 \vee a = A_2 \ldots)$$
$$\vee (F_i(s) \wedge (a \neq A_3) \wedge (a \neq A_4) \ldots)$$

made it so

exclude.

- Several actions that make the fluent true and several that make the fluent false: Formalize using the predicate $PosEffect(a, F_i)$ and $NegEffect(a, F_i)$.

$Poss(a, s) \rightarrow$

$$F_i(Result(a, s)) \leftrightarrow PosEffect(a, F_i)$$
$$\vee [F_i(s) \wedge \neg NegEffect(a, F_i)]$$

$PosEffect(A_1, F_i), PosEffect(A_1, F_i)$

$NegEffect(A_3, F_i), NegEffect(A_4, F_i)$

* This can be done efficiently: get current action, and fetch its effects, then update those fluents $O(Et)$.

So you want to exclude them

Solving the Inferential Frame Problem

- Typical frame axiom: $Poss(a, s) \rightarrow$

$$F_i(Result(a, s)) \leftrightarrow (a = A_1 \vee a = A_2 \ldots)$$
$$\vee (F_i(s) \wedge (a \neq A_3) \wedge (a \neq A_4) \ldots)$$

- Several actions that make the fluent true and several that make the fluent false: Formalize using the predicate $PosEffect(a, F_i)$ and $NegEffect(a, F_i)$.

$Poss(a, s) \rightarrow$

$$F_i(Result(a, s)) \leftrightarrow PosEffect(a, F_i)$$
$$\vee [F_i(s) \wedge \neg NegEffect(a, F_i))]$$

$PosEffect(A_1, F_i), PosEffect(A_1, F_i)$

$NegEffect(A_3, F_i), NegEffect(A_4, F_i)$

* This can be done efficiently: get current action, and fetch its effects, then update those fluents $O(Et)$.

So that corresponds to this

Solving the Inferential Frame Problem

- Typical frame axiom: $Poss(a, s) \rightarrow$

$$F_i(Result(a, s)) \leftrightarrow (a = A_1 \vee a = A_2 \ldots) \vee (F_i(s) \wedge (a \neq A_3) \wedge (a \neq A_4) \ldots)$$

- Several actions that make the fluent true and several that make the fluent false: Formalize using the predicate $PosEffect(a, F_i)$ and $NegEffect(a, F_i)$.

$$Poss(a, s) \rightarrow$$

$$F_i(Result(a, s)) \leftrightarrow PosEffect(a, F_i) \vee [F_i(s) \wedge \neg NegEffect(a, F_i)]$$

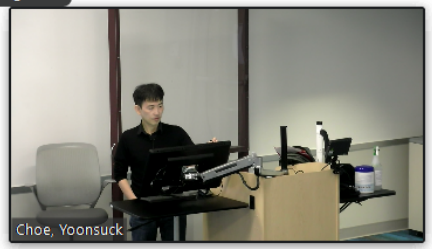
$$PosEffect(A_1, F_i), PosEffect(A_1, F_i)$$

$$NegEffect(A_3, F_i), NegEffect(A_4, F_i)$$

already true.

* This can be done efficiently: get current action, and fetch its effects, then update these fluents $O(Lt)$.

Do you want to exclude them? Well, making sure that this flew on was already true.



Solving the Inferential Frame Problem

- Typical frame axiom: $Poss(a, s) \rightarrow$

$$F_i(Result(a, s)) \leftrightarrow (a = A_1 \vee a = A_2 \ldots) \vee (F_i(s) \wedge (a \neq A_3) \wedge (a \neq A_4) \ldots)$$

- Several actions that make the fluent true and several that make the fluent false: Formalize using the predicate $PosEffect(a, F_i)$ and $NegEffect(a, F_i)$.

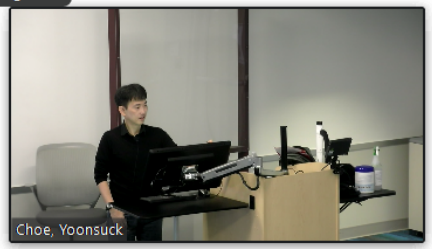
$$Poss(a, s) \rightarrow F_i(Result(a, s)) \leftrightarrow PosEffect(a, F_i) \vee [F_i(s) \wedge \neg NegEffect(a, F_i)]$$

$PosEffect(A_1, F_i), PosEffect(A_2, F_i)$ typo.

$NegEffect(A_3, F_i), NegEffect(A_4, F_i)$

* This can be done efficiently: get current action, and fetch its effects, then update these fluents $O(Lt)$.

A 2 is another action that would result in a positive effect for fi, and so on.

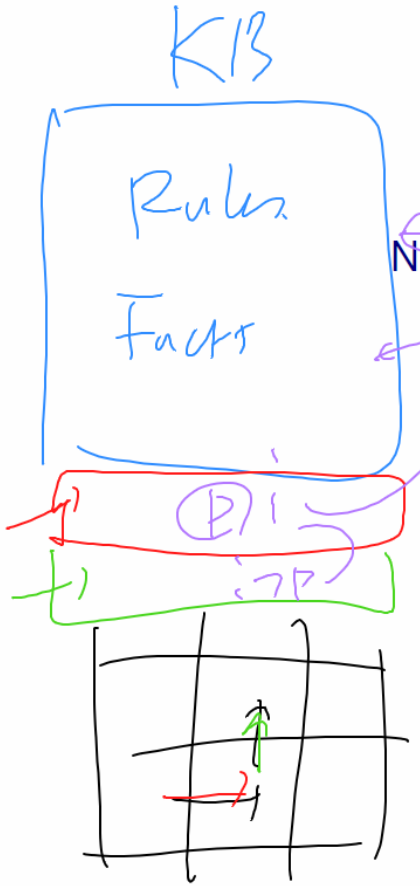


Other Formalisms

- Event calculus: Fluents hold at diffetent time points, not situations. Reasoning is done over time.
- Other constructs: generalized events (spatiotemporal), process, intervals, etc.
- Formal theory of belief: propositional attitude, reification, etc.

Skip

But well, these are just for your reference let's say just we skip the



Truth Maintenance Systems

New facts inferred from the KB can turn out to be incorrect.

- Let's say P was derived in the KB and later it was found that $\neg P$.
- Adding $\neg P$ to the KB will invalidate the entire KB, so P should be removed ($Retract(KB, P)$).
- Care needs to be taken since other facts in the KB may have been derived from P, etc.
- Truth maintenance systems are designed to handle these complications.

3. Apply $\{x/f(y), y/B\}$ to $P(x, y) \vee Q(x)$.

Problem 5 (12 pts): For each of the following, (1) find the unifier, and (2) show the unified expression. For example, given $P(A)$ and $P(x)$, the unifier would be $\{x/A\}$, and the unified expression $P(A)$. If the pair of expressions is not unifiable, indicate so. (3 points each). Note: Show all steps, including the disagreement set D_k for each iteration, and the σ_k.

1. $P(x, f(B))$ and $P(A, f(y))$
2. $P(y, y)$ and $P(x, f(A))$
3. $P(f(y), x, y)$ and $P(f(g(w)), w, g(A))$
4. $P(A, f(y), y, A)$ and $P(x, f(g(x)), g(B), w)$

1.3 Proof by resolution

Problem 6 (16 pts): (1) Translate the following into first-order logic, (2) convert the resulting formulas into a canonical form, and (3) prove the theorem using resolution.

Given:

1. If a child is young, then he/she is adored.
2. Every child who is adored and fed well is happy.
3. Every child is fed well.

Show that the following is a logical consequence of the above:

4. (Conclusion) Every young child is happy.

Planning Approaches

- State-space search: forward or backward.
- Heuristic search: subgoal independence assumption.
- Partial-order planning: utilize problem decomposition. Can place two actions into a plan without specifying the order. Several different total order plans can be constructed from partial order plans.

Using this kind of partial between

Acquisition of New Information and Probability

- The degree of belief changes as an agent perceives or acquires new information from the world: we call this the **evidence**.
- This is analogous to saying whether or not a given logical sentence is entailed by (i.e. is a logical consequence of) the knowledge base, because the truth value can change when new facts are added to the KB.
- Before the evidence is received, we talk about **prior** or **unconditional** probability.
- After the evidence is obtained, we talk about **posterior** or **conditional** probability.

Acquisition of New Information and Probability

- The degree of belief changes as an agent perceives or acquires new information from the world: we call this the **evidence**.
- This is analogous to saying whether or not a given logical sentence is entailed by (i.e. is a logical consequence of) the knowledge base, because the truth value can change when new facts are added to the KB.
- Before the evidence is received, we talk about **prior** or **unconditional** probability.
- After the evidence is obtained, we talk about **posterior** or **conditional** probability.

So this is either called conditional, or the poststeria prime p

Acquisition of New Information and Probability

- The degree of belief changes as an agent perceives or acquires new information from the world: we call this the **evidence**.
- This is analogous to saying whether or not a given logical sentence is entailed by (i.e. is a logical consequence of) the knowledge base, because the truth value can change when new facts are added to the KB.
- Before the evidence is received, we talk about **prior** or **unconditional** probability.
- After the evidence is obtained, we talk about **posterior** or **conditional** probability.

Acquisition of New Information and Probability

- The degree of belief changes as an agent perceives or acquires new information from the world: we call this the **evidence**.
- This is analogous to saying whether or not a given logical sentence is entailed by (i.e. is a logical consequence of) the knowledge base, because the truth value can change when new facts are added to the KB.
- Before the evidence is received, we talk about **prior** or **unconditional** probability.
- After the evidence is obtained, we talk about **posterior** or **conditional** probability.

So you can keep on adding, and more

Rational Decisions Under Uncertainty: Decision Theory

- There are trade-offs, and an agent must first have **preferences** between different results when a certain plan was executed.
- **Utility theory** deals with such preferences: how useful is such and such result to the agent?
- **Decision theory** is a general theory of rational decision under uncertainty, combining **probability theory** and **utility theory**.

Theory and utility Theory to come up with, like a rational way to make decisions, to pick which action in a certain given

Decision Theory

- An agent is rational iff it chooses the action that yields the highest expected utility, averaged over all possible outcomes of the action: **Principle of Maximum Expected Utility**
- Example: backgammon (discussed earlier) – min-max trees with probabilistic levels.

What we want to do is to use the principle of Mac maximum expected utility

Decision Theoretic Agent

function DT-Agent (*percept*) **returns** *action*

static: a set probabilistic *belief* about the state of the world

calculate updated probabilities for current state based on *percept* and past actions

calculate outcome probabilities for actions, given action descriptions and prob of current states.

select *action* with highest expected utility given prob of outcomes and utility information.

return *action*

And so on

Decision Theoretic Agent

function DT-Agent (*percept*) **returns** *action*

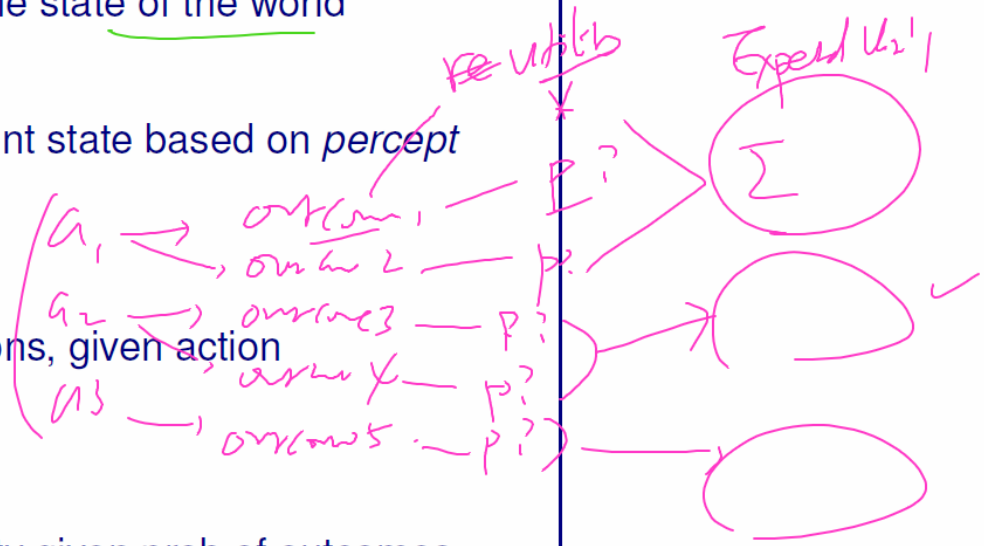
static: a set probabilistic *belief* about the state of the world

calculate updated probabilities for current state based on *percept* and past actions

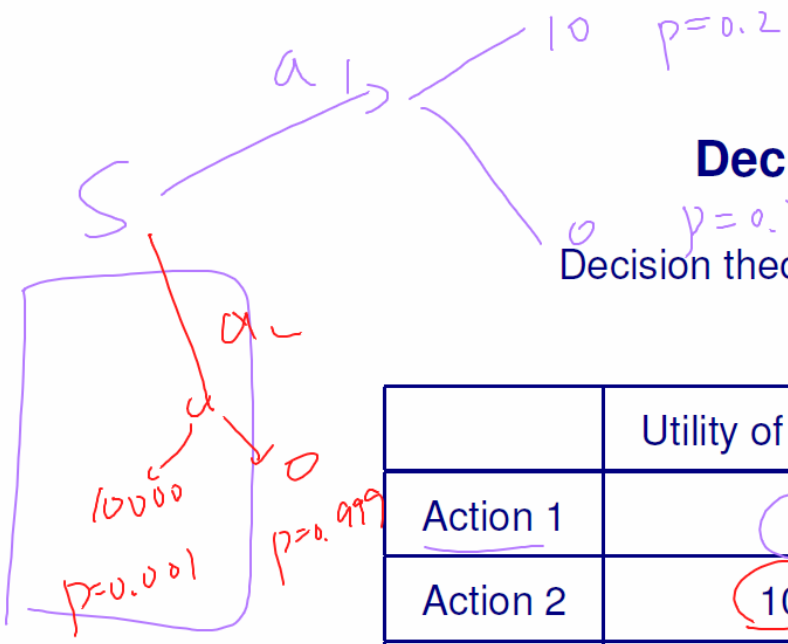
calculate outcome probabilities for actions, given action descriptions and prob of current states.

select *action* with highest expected utility given prob of outcomes and utility information.

return *action*



So this is



Decision Theory: Example

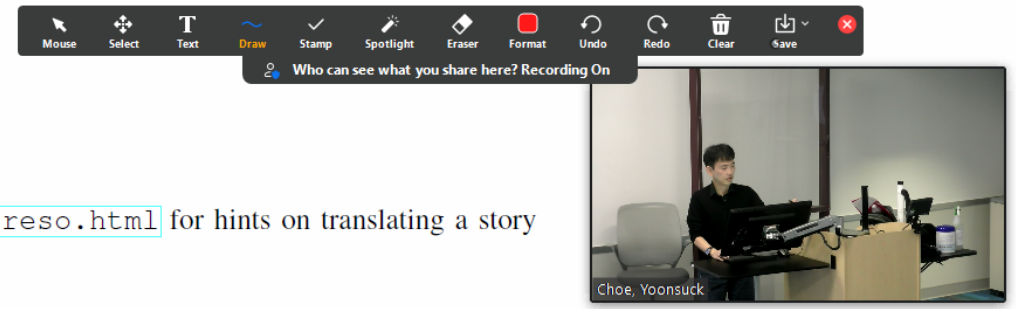
Decision theory = Probability theory + Utility theory

	Utility of Resulting State	Probability
Action 1	10 or 0	0.2, 0.8, respectively
Action 2	10000 or 0	0.001, 0.999, respectively
Action 3	5 or 0	0.799, 0.201, respectively

Which action would an optimal Decision Theoretic Agent take?

Of 0. then this has a probability of 0 point 0 0 one, and probably t of your point 9, 9, 9, and so on, and same fractio

- $Adored(x)$: child x is adored.
- $Fed(x)$: child x is fed well.
- $Happy(x)$: child x is happy.



See `https://www.cs.utexas.edu/users/novak/reso.html` for hints on translating a story into FOL and proving it using resolution.

1.4 Programming

Problem 7 (25 pts): ~~Unification algorithm in Python~~.

For this programming component, you will be implementing a simple resolution-based theorem prover in Python.

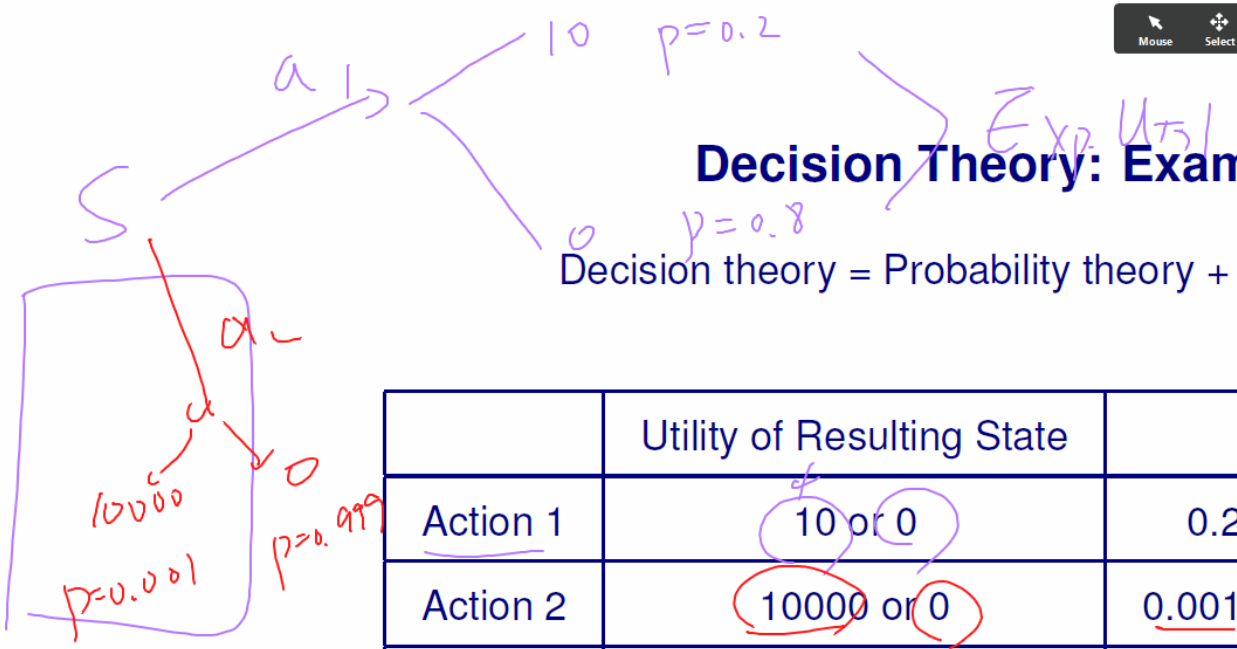
- Download the `resolution.ipynb` file from Canvas→Files→Assignment.
- Then upload to `https://research.colab.google.com` (use your TAMU NetID to login), and run it.
- Look at the test code to get yourself familarized with the provided facilities, such as `resolve()`.
- You can create new code cells to experiment.

for propositional logic

The only part you need to implement is the `prover()` function. You will be implementing the *two-pointer method* (same as level saturation), using the *set-of-support strategy*.

Here's a description of the algorithm[1]:

1. Initialize the two pointers (simply, the clause index) so that you will be using the set-of-support strategy. This can be done in the following way.
   - Set the "inner loop" pointer to the front of the list of clauses.
   - Set the "outer loop" pointer to the first clause resulting from the negated conclusion.
   - Go to Step 2.

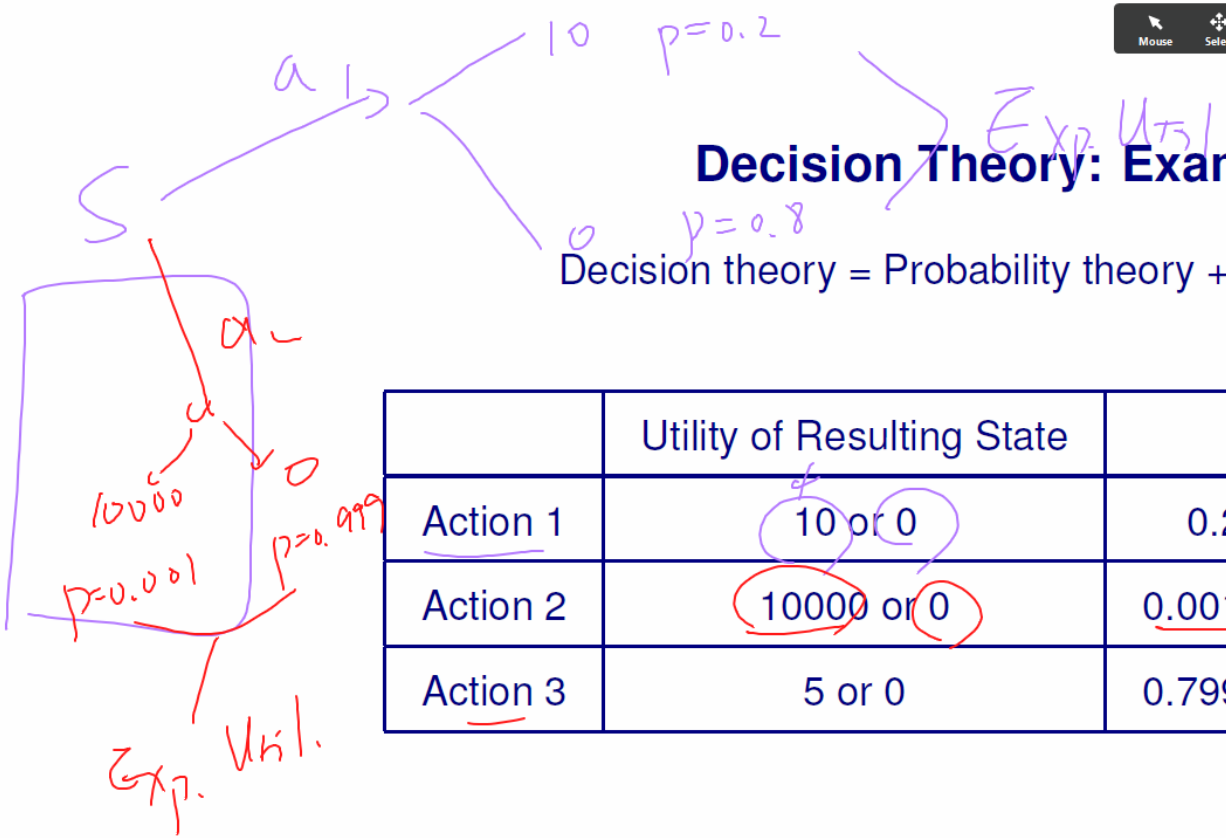


Decision Theory: Example

Decision theory = Probability theory + Utility theory

	Utility of Resulting State	Probability
Action 1	10 or 0	0.2, 0.8, respectively
Action 2	10000 or 0	0.001, 0.999, respectively
Action 3	5 or 0	0.799, 0.201, respectively

Which action would an optimal Decision Theoretic Agent take?

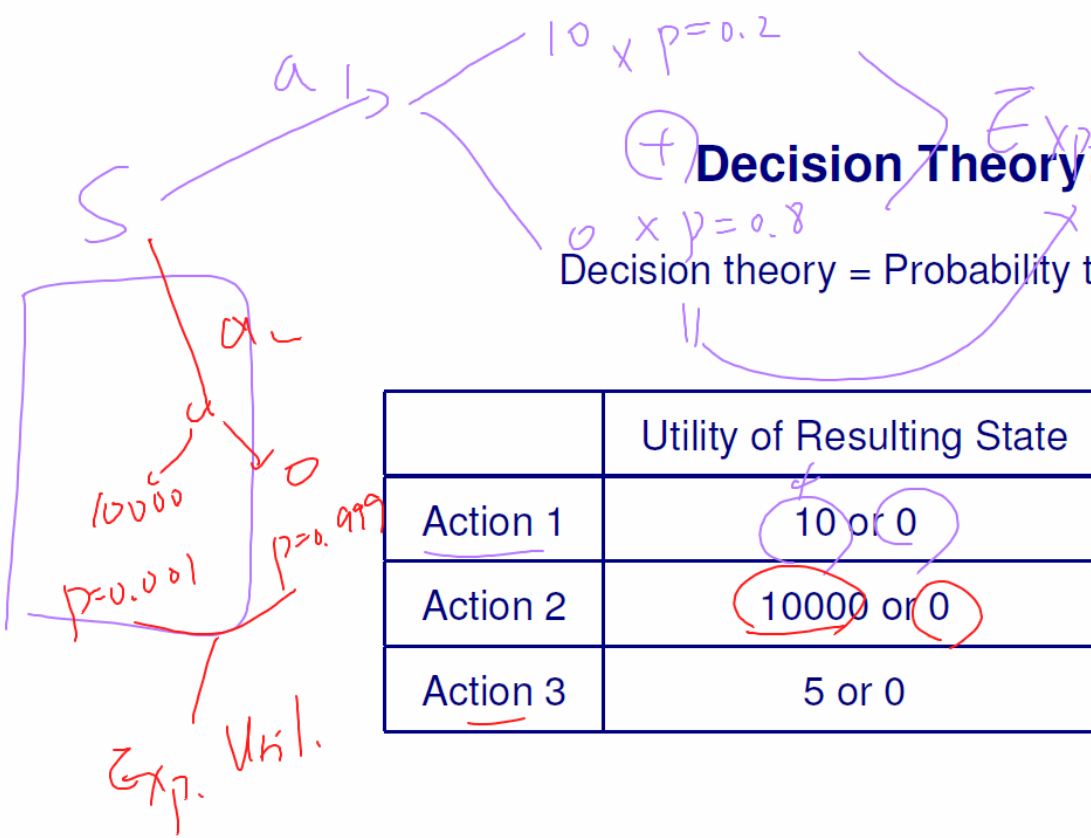


Decision Theory: Example

Decision theory = Probability theory + Utility theory

	Utility of Resulting State	Probability
Action 1	10 or 0	0.2, 0.8, respectively
Action 2	10000 or 0	0.001, 0.999, respectively
Action 3	5 or 0	0.799, 0.201, respectively

Which action would an optimal Decision Theoretic Agent take?



Decision Theory: Example

Decision theory = Probability theory + Utility theory

	Utility of Resulting State	Probability
Action 1	10 or 0	0.2, 0.8, respectively
Action 2	10000 or 0	0.001, 0.999, respectively
Action 3	5 or 0	0.799, 0.201, respectively

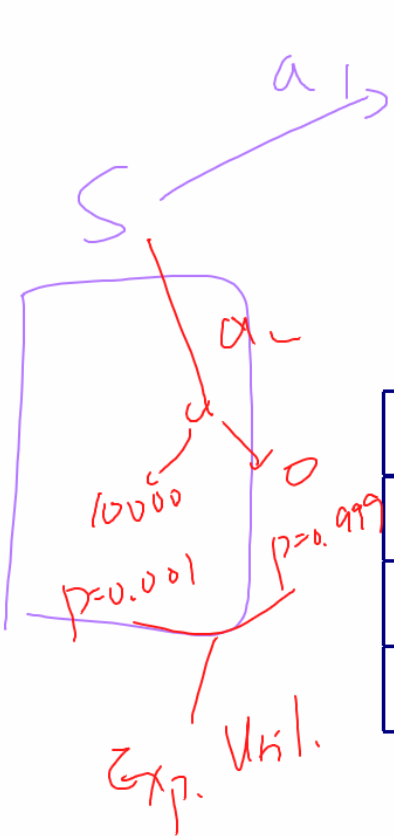
Which action would an optimal Decision Theoretic Agent take?

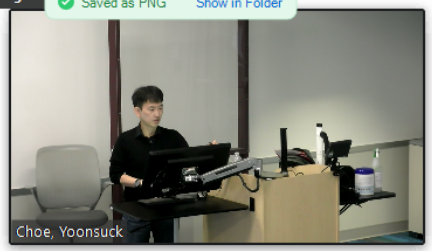
Decision Theory: Example

Decision theory = Probability theory + Utility theory

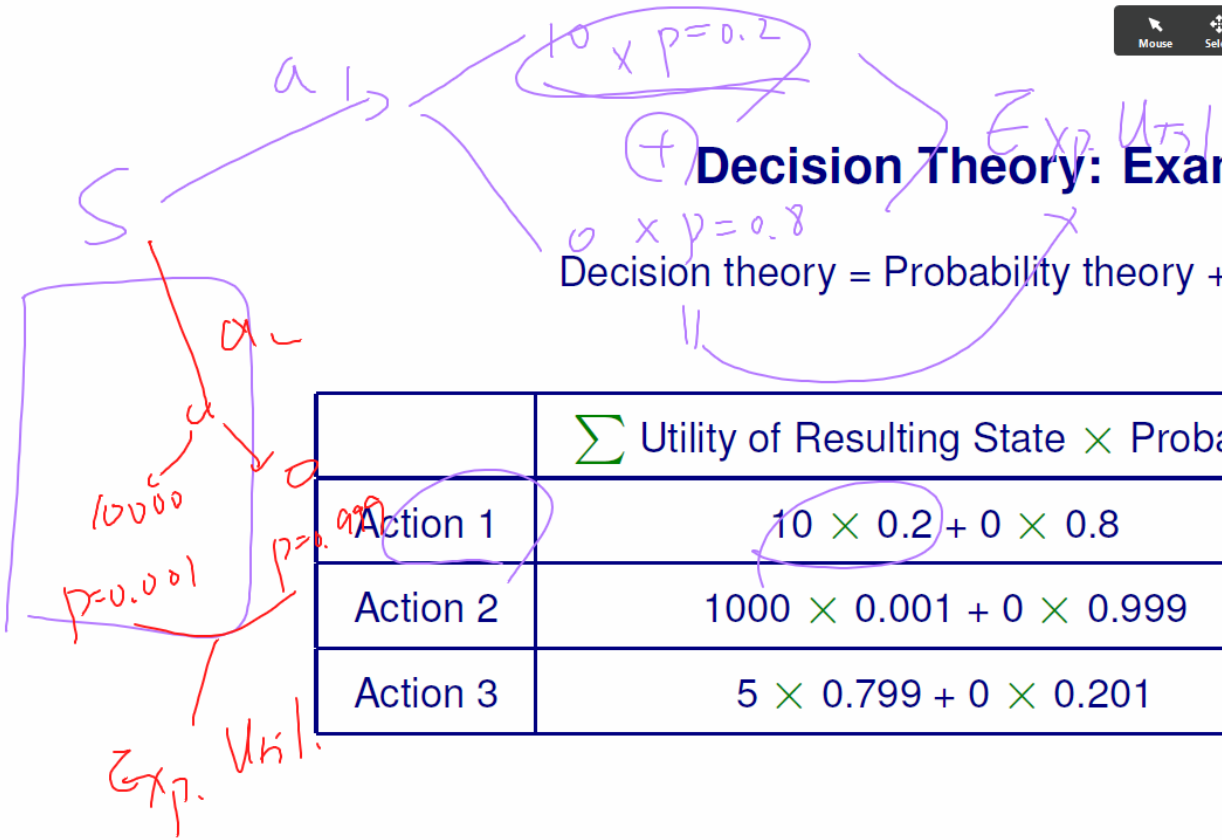
	Utility of Resulting State	Probability
Action 1	10 or 0	0.2, 0.8, respectively
Action 2	10000 or 0	0.001, 0.999, respectively
Action 3	5 or 0	0.799, 0.201, respectively

Which action would an optimal Decision Theoretic Agent take?





So basically it'd be this times that plus this time is that that'd be the speed utility and same for the action to any

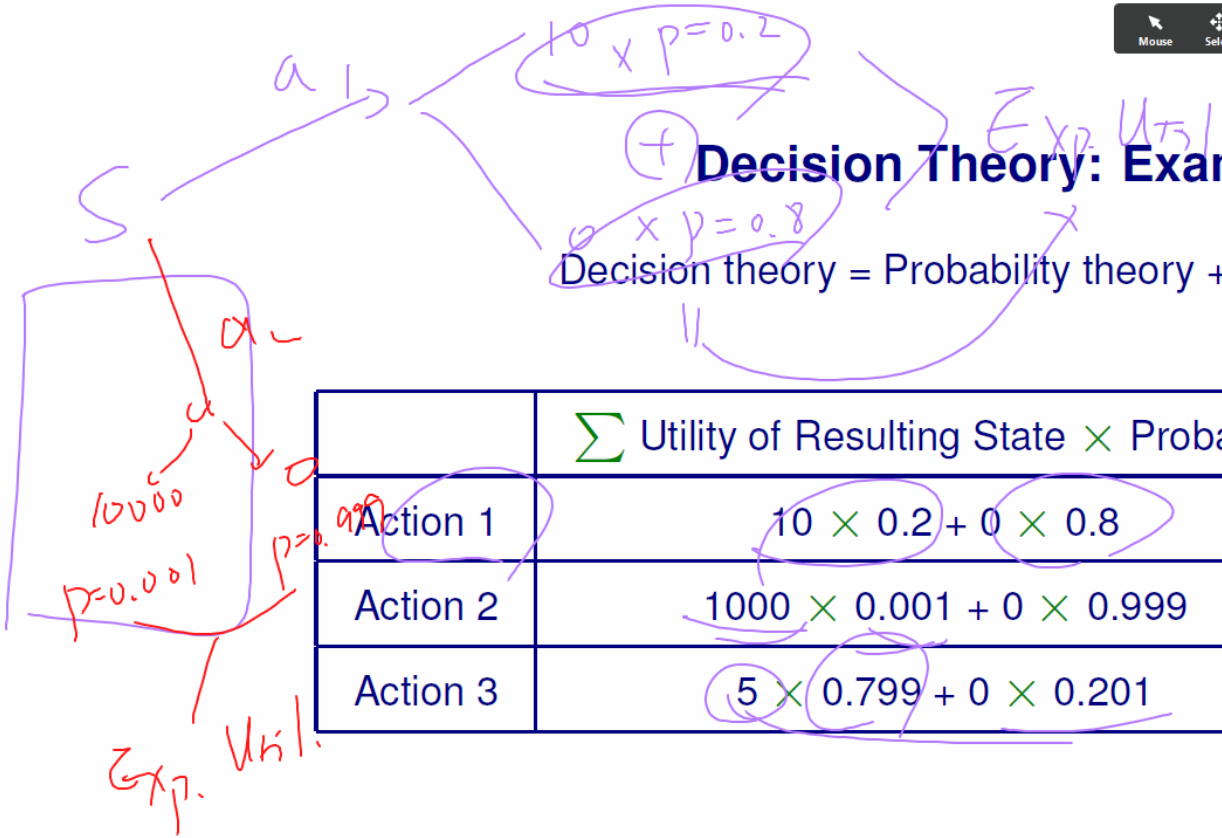


Decision Theory: Example

Decision theory = Probability theory + Utility theory

	$\sum$ Utility of Resulting State $\times$ Probability	Expected Utility
Action 1	$10 \times 0.2 + 0 \times 0.8$	2
Action 2	$1000 \times 0.001 + 0 \times 0.999$	1
Action 3	$5 \times 0.799 + 0 \times 0.201$	3.995 $\longleftarrow$

Action 3 has the maximum expected utility, thus action 3 will be carried out.

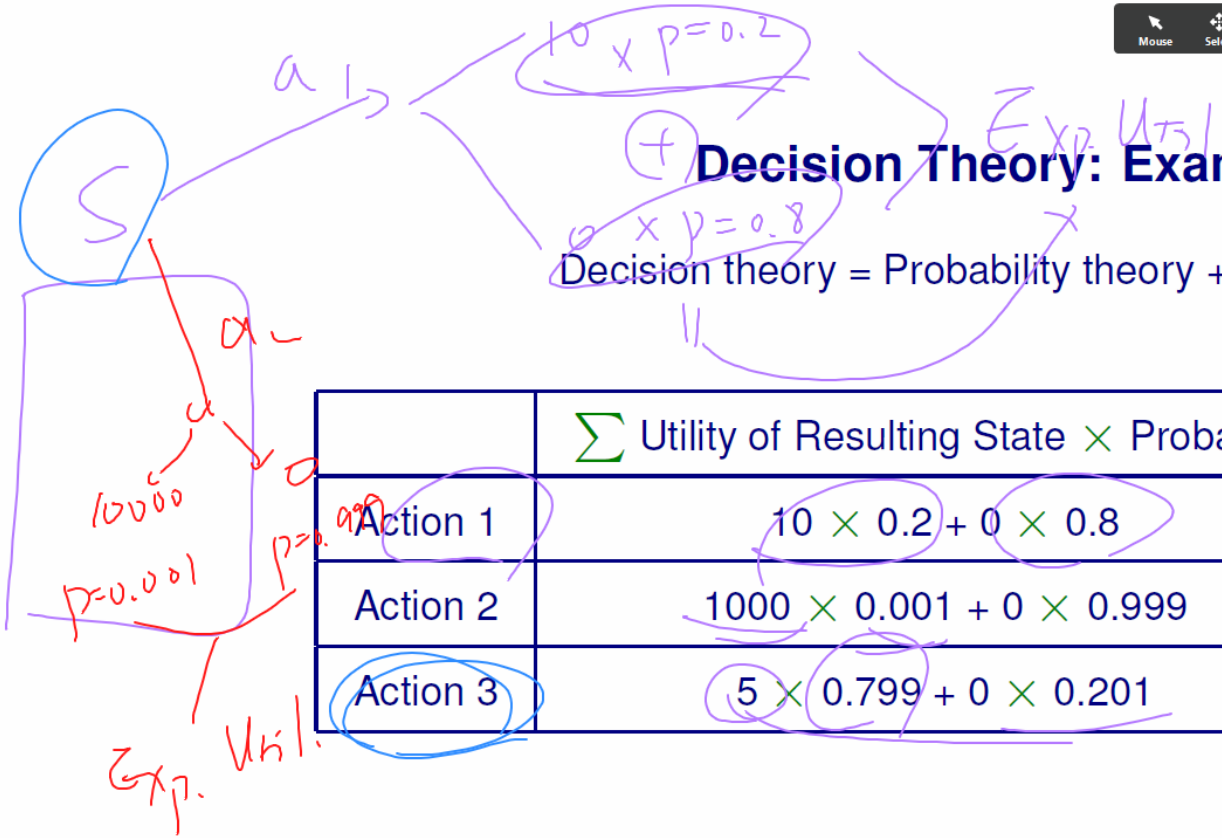


Decision Theory: Example

Decision theory = Probability theory + Utility theory

	$\sum$ Utility of Resulting State $\times$ Probability	Expected Utility
Action 1	$10 \times 0.2 + 0 \times 0.8$	2
Action 2	$1000 \times 0.001 + 0 \times 0.999$	1
Action 3	$5 \times 0.799 + 0 \times 0.201$	3.995

Action 3 has the maximum expected utility, thus action 3 will be carried out.



Decision Theory: Example

Decision theory = Probability theory + Utility theory

	$\sum$ Utility of Resulting State $\times$ Probability	Expected Utility
Action 1	$10 \times 0.2 + 0 \times 0.8$	2
Action 2	$1000 \times 0.001 + 0 \times 0.999$	1
Action 3	$5 \times 0.799 + 0 \times 0.201$	3.995

Action 3 has the maximum expected utility, thus action 3 will be carried out.

Homework 3 | Files | resolution.ipynb - Colaboratory

colab.research.google.com/drive/1yORqeLdFOgLqCDKnoekwl6k5CB5dpk8i

22 FALL CSCE 420 5... | 22 FALL CSCE 633 6... | Video Conferencing...

Mouse | Select | Text | Draw | Stamp | Spotlight | Eraser | Format | Undo | Redo | Clear | Save

Who can see what you share here? Recording On

resolution.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:22 AM

Comment | Share

+ Code + Text


```
clause representation:

  [<positive-proposition-list>, <negative-proposition-list>]

proposition-list representation:

  [ 'p', 'q', 'r' ]

  Note: 'p', 'q', 'r', 's' are propositions : CASE SENSITIVE!

Example clause:

  [['p', 'q'] ['r']] : This is clause P \/ Q \/ ~R
'''

DEBUG=True

#-----------------------------------------------------------------
def rm_item(lst, item):
#-----------------------------------------------------------------
  '''
  function: rm_pred()
```

pos neg

resolution.ipynb | hw3-22fall.pdf | slide05-prob.pdf

They put all the narrative it was in the second list list, and that's the close

You are screen sharing | Stop Share | Show all

Duchess of Cornwall | Prince of Wales | Princess of Wales

Homework 3 | Files | resolution.ipynb - Colaboratory

colab.research.google.com/drive/1yORqeLdFOgLqCDKnoekwl6k5CB5dpk8i

22 FALL CSCE 420 5... | 22 FALL CSCE 633 6... | Video Conferencing...

Mouse | Select | Text | Draw | Stamp | Spotlight | Eraser | Format | Undo | Redo | Clear | Save

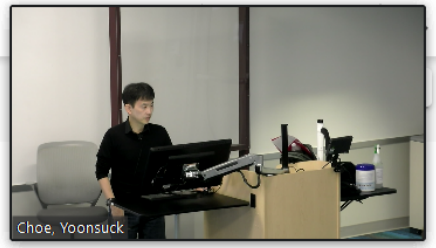
Who can see what you share here? Recording On

resolution.ipynb

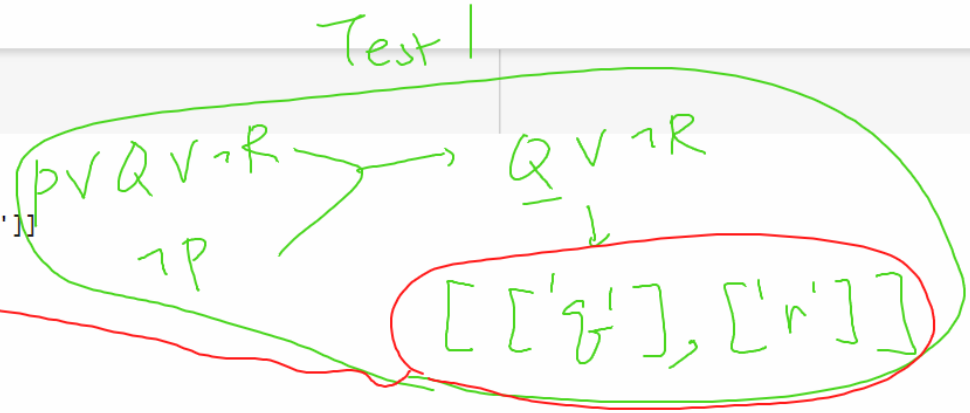
File Edit View Insert Runtime Tools Help Last saved at 11:22 AM

Comment | Share

+ Code + Text




```
print("null? "+str(null_p(test5)))
```




```
test1
resolving: [['p', 'q'], ['r']] and [[], ['p']]
resolvent = [['q'], ['r']]
null? False

test2
resolving: [['p'], []] and [[], ['p']]
resolvent = [[], []]
null? True

test3
resolving: [['p', 'q'], ['r']] and [['r'], ['p']]
resolvent = False
null? False

test4
resolving: [['p', 'q'], ['r']] and [['r'], []]
resolvent = [['p', 'q'], []]
null? False

test5
resolving: [['p', 'q'], ['r']] and [['r', 'p'], []]
```

resolution.ipynb | hw3-22fall.pdf | slide05-prob.pdf

No that's true so on so it's pretty

Duchess of Cornwall | Prince of Wales | Princess of Wales

You are screen sharing | Stop Share

Show all

Homework 3 | Files | resolution.ipynb - Colaboratory

colab.research.google.com/drive/1yORqeLdFOgLqCDKnoekwl6k5CB5dpk8i

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

22 FALL CSCE 420 5... 22 FALL CSCE 633 6... Video Conferencing...

resolution.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:22 AM

Comment Share

+ Code + Text


```
theorem2 = [ ]  # enter the example theorm in the homework

theorem3 = [ ]  # enter your own example for testing


#-------------------------------------------------------
# prover() function : you must implement this function
#-------------------------------------------------------
def prover(thm, goal):
#-------------------------------------------------------------
  '''
  function prover: implement this

  two-pointer method, with set of support
  - arguments:
    thm : theorem
    goal : integer index (clause number where the negated conclusion starts)
  - show resolution steps
  - if null_p checks, return True (theorem proven)
  - otherwise return False
  '''

  print('\nprover():\n\nTheorem:')
```

For "set of support".

resolution.ipynb hw3-22fall.pdf slide05-prob.pdf

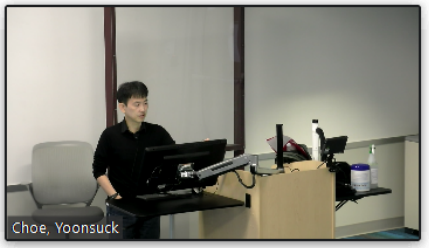
Close right to use the set of support. approach

You are screen sharing Stop Share

Show all

Duchess of Cornwall Prince of Wales Princess of Wales

- Move the "inner loop" pointer forward one clause.
- If the "outer loop" pointer goes beyond the last clause in the list, stop; the theorem cannot be proved. Return "failure".
- If the "inner loop" pointer has not reached the "outer loop" pointer, go to the Resolve step (Step 2).
- Otherwise, reset the "inner loop" pointer to the front of the list of clauses and move the "outer loop" pointer forward one clause, then go to the Resolve step (Step 2).



Testing requirement:

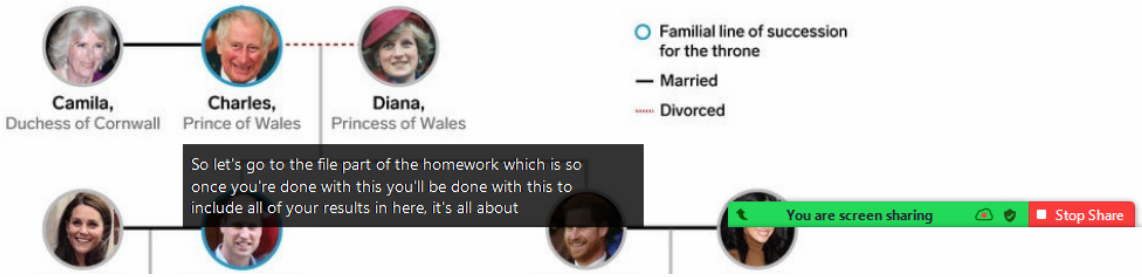
1. Test the `prover()` function on the default theorem in the skeleton code.
2. Enter the Unicorn theorem in slide03-logic.pdf page 41 into your code as a new theorem. Test all three possible outcomes (M or G or H: try one conclusion at a time), given the premises (under "Given"). Don't forget to negate the conclusion.
3. In all example cases, report how many clauses were generated before deriving False (empty clause), or failing.

Include the following in one zip file:

1. `resolution.ipynb`: this should include all your code and test runs.

Problem 8 (10 pts): Simple knowledge base in Prolog.

Consider the following family tree (British royal family).



Frame Problem

- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

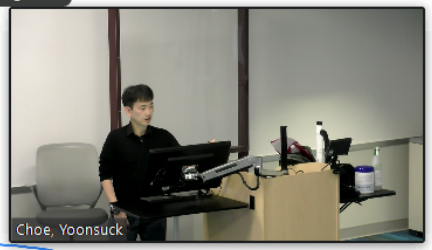
$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0))$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken*.
- Initial solution: *Frame axioms*

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s)$$
$$\rightarrow At(o,x,Result(Go(y,z),s)).$$

This says moving does not affect the gold when it is not held. Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).

So you start from s. 0, and then you raise this



Frame Problem

- In the previous slide, we cannot deduce if the following can be proven (G_1 represents a particular lump of gold):

$$At(G_1, [1,1], Result([Go([1,1],[1,2]), Grab(G_1), Go([1,2],[1,1])], S_0))$$

- It is because the effect axioms say only *what should change*, but not *what does not change when actions are taken.*
- Initial solution: *Frame axioms*

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s)$$
$$\rightarrow At(o,x,Result(Go(y,z),s)).$$

This says moving does not affect the gold when it is not held. Problem is that you need $O(AF)$ such axioms for all $(action, fluent)$ pair (A: num of actions, F: num of fluent predicates).

One. The location of G, one is 1 one in situation 3, whereas the situation 3 is basically this situation