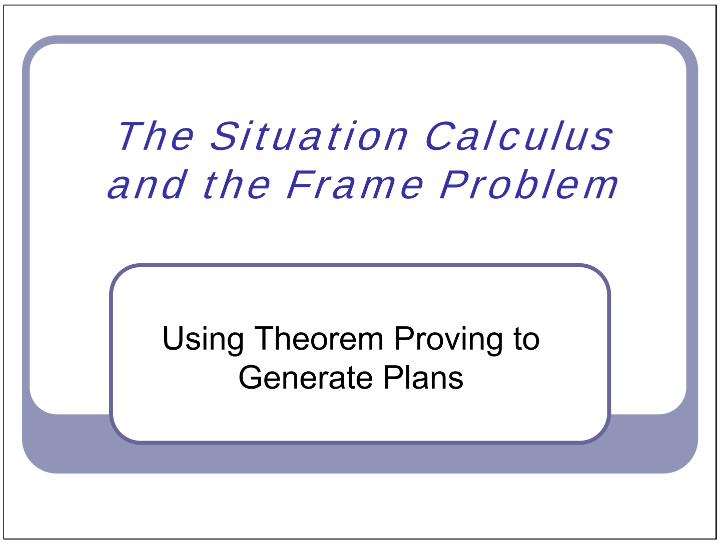
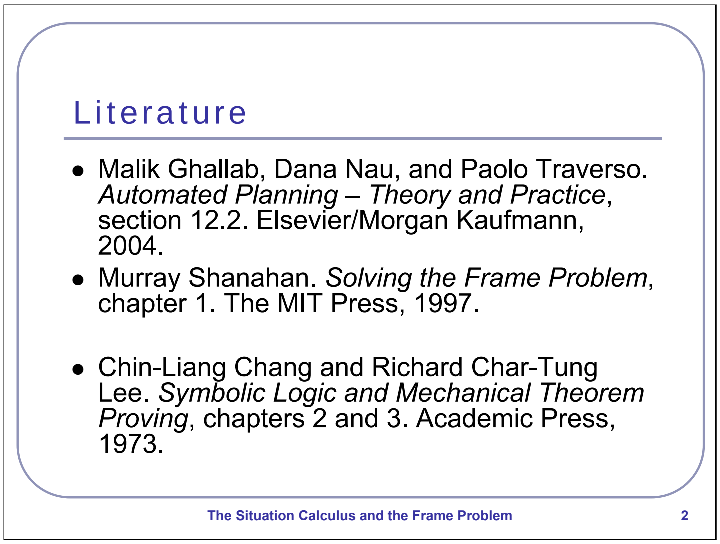


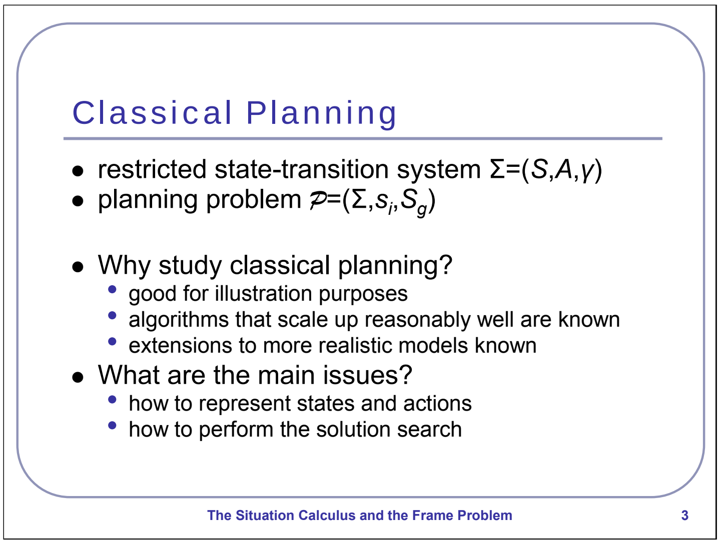
The Situation Calculus and the Frame Problem

•Using Theorem Proving to Generate Plans



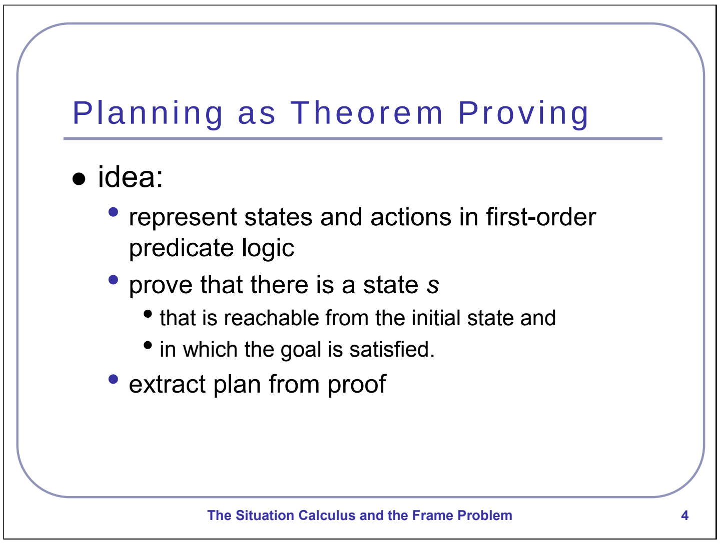
Literature

- **Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning – Theory and Practice*, section 12.2. Elsevier/Morgan Kaufmann, 2004.**
- **Murray Shanahan. *Solving the Frame Problem*, chapter 1. The MIT Press, 1997.**
- **Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*, chapters 2 and 3. Academic Press, 1973.**
    - for propositional and first-order logic



Classical Planning

- **restricted state-transition system $\Sigma=(S,A,\gamma)$**
  - finite, fully observable, deterministic, and static with restricted goals and implicit time
- **planning problem $\mathcal{P}=(\Sigma,s_i,S_g)$**
  - task of planning: synthesize (offline) the sequence of actions that is a solution
- **Why study classical planning?**
  - **good for illustration purposes**
  - **algorithms that scale up reasonably well are known**
  - **extensions to more realistic models known**
- **What are the main issues?**
  - **how to represent states and actions**
    - domain-independence, avoid enumerating S and γ, avoiding the frame problem
  - **how to perform the solution search**
    - efficient search



Planning as Theorem Proving

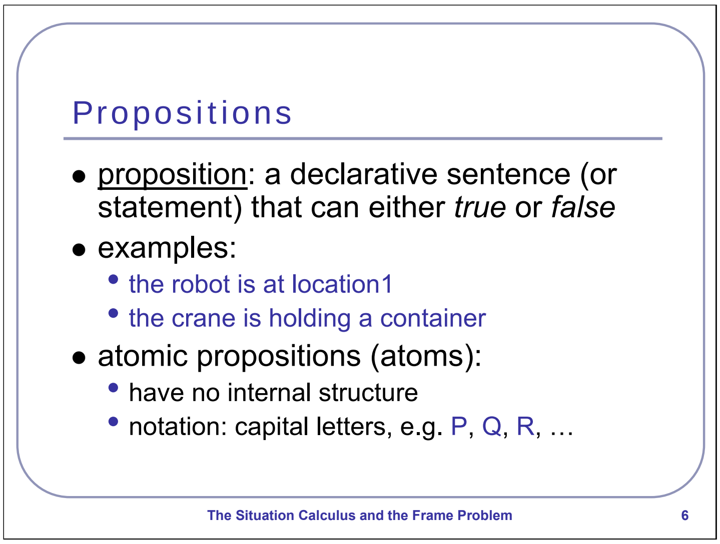
- **idea:**
  - **represent states and actions in first-order predicate logic**
  - **prove that there is a state *s***
    - **that is reachable from the initial state and**
    - **in which the goal is satisfied.**
    - proof must be constructive
  - **extract plan from proof**
- all reasoning done by theorem prover, only problem is representation

Overview

- Propositional Logic
- First-Order Predicate Logic
- Representing Actions
- The Frame Problem
- Solving the Frame Problem

Overview

- **Propositional Logic**
  - now: a very simple formal logic
- **First-Order Predicate Logic**
- **Representing States and Actions**
- **The Frame Problem**
- **Solving the Frame Problem**



Propositions

•**<u>proposition</u>: a declarative sentence (or statement) that can either *true* or *false***

•**examples:**

- **the robot is at location1**
- **the crane is holding a container**

•**atomic propositions (atoms):**

- **have no internal structure** – “robot is at location1” unrelated to “robot is at location2”
- **notation: capital letters, e.g. P, Q, R, …**

Well-Formed Formulas

- an atom is a formula
- if G is a formula, then (¬G) is a formula
- if G and H are formulas, then (G∧H), (G∨H), (G→H), (G↔H) are formulas.
- all formulas are generated by applying the above rules
- logical connectives: ¬, ∧, ∨, →, ↔

The Situation Calculus and the Frame Problem 7

Well-Formed Formulas

•an atom is a formula

•if G is a formula, then (¬G) is a formula

•if G and H are formulas, then (G∧H), (G∨H), (G→H), (G↔H) are formulas.

•all formulas are generated by applying the above rules

•logical connectives:

- **•¬:** "not", negation
- **•∧:** "and", conjunction
- **•∨:** "or", disjunction
- **•→:** "implies", implication
- **•↔:** "if and only if", co-implication, equivalence

Truth Tables

G	H	$\neg G$	$G \wedge H$	$G \vee H$	$G \rightarrow H$	$G \leftrightarrow H$
true	*true*	*false*	*true*	*true*	*true*	*true*
true	*false*	*false*	*false*	*true*	*false*	*false*
false	*true*	*true*	*false*	*true*	*true*	*false*
false	*false*	*true*	*false*	*false*	*true*	*true*

Truth Tables

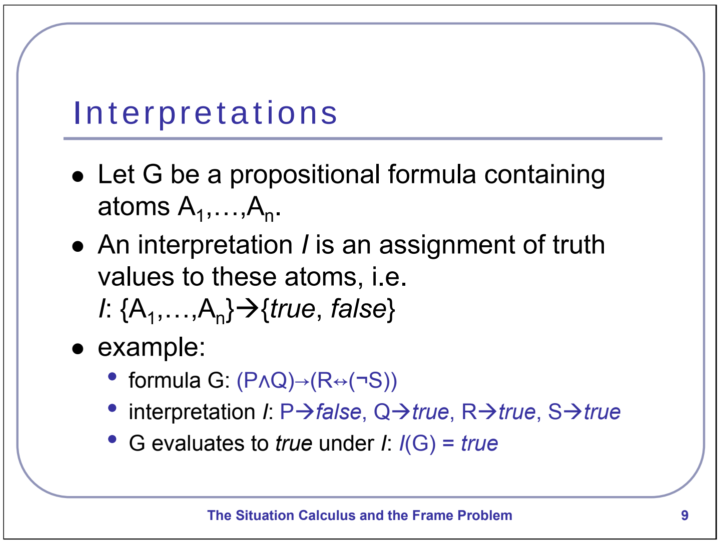
•**$\neg G$:** true if and only if G is false

•**$G \wedge H$:** true if and only if both, G and H are true

•**$G \vee H$:** true if and only if one or both of G or H is true (not exclusive)

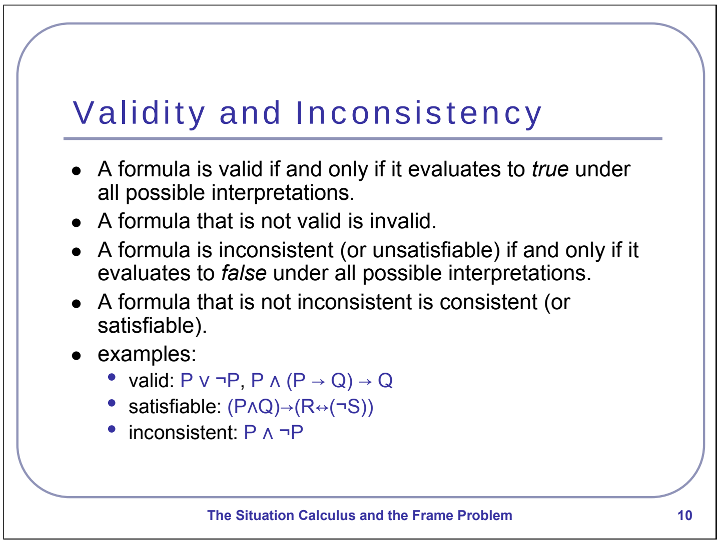
•**$G \rightarrow H$:** true if and only if G is false or H is true (or both)

•**$G \leftrightarrow H$:** true if and only if G and H have the same truth vlaue



Interpretations

- **Let G be a propositional formula containing atoms $A_1,\ldots,A_n$.**
- **An interpretation *I* is an assignment of truth values to these atoms, i.e.**
  ***I*: $\{A_1,\ldots,A_n\} \rightarrow \{$*true*, *false*$\}$**
- **example:**
  - **formula G: $(P \wedge Q) \rightarrow (R \leftrightarrow (\neg S))$**
  - **interpretation *I*: P→*false*, Q→*true*, R→*true*, S→*true***
  - **G evaluates to *true* under *I*: *I*(G) = *true*** (use truth tables on previous slide to evaluate)
  - see also http://www.aiai.ed.ac.uk/~gwickler/truth-table.html



Validity and Inconsistency

•A formula is valid if and only if it evaluates to *true* under all possible interpretations.

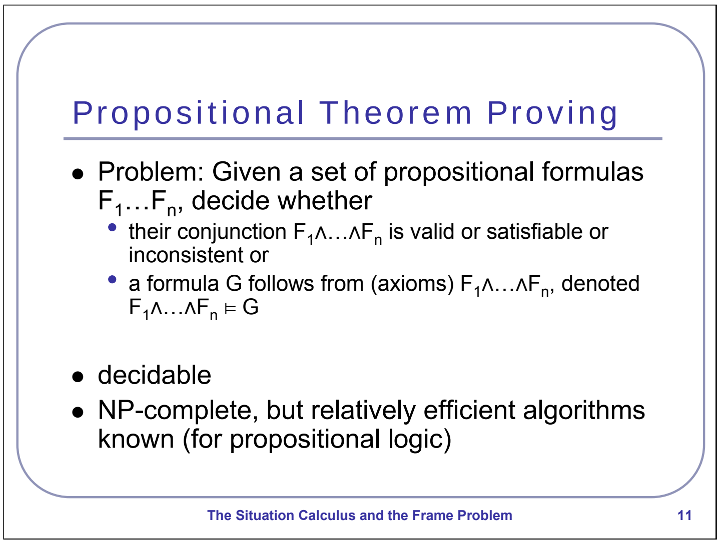
•A formula that is not valid is invalid. - *false* under at least one interpretation, but may be *true* under others

•A formula is inconsistent (or unsatisfiable) if and only if it evaluates to *false* under all possible interpretations.

•A formula that is not inconsistent is consistent (or satisfiable). - *true* under at least one interpretation, but may be *false* under others or may be valid

•examples:

- **•valid: $P \vee \neg P$** (excluded third)**, $P \wedge (P \rightarrow Q) \rightarrow Q$** (modus ponens)
- **•satisfiable: $(P \wedge Q) \rightarrow (R \leftrightarrow (\neg S))$**
- **•inconsistent: $P \wedge \neg P$**



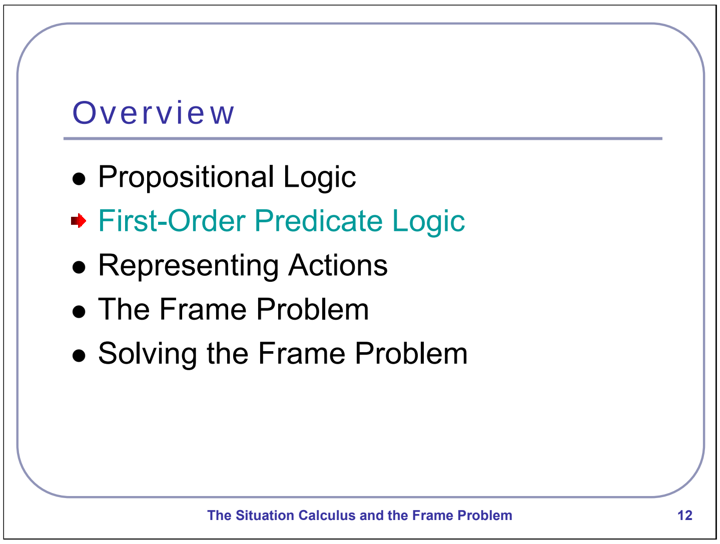
Propositional Theorem Proving

•**Problem: Given a set of propositional formulas $F_1 \ldots F_n$, decide whether**

- •**their conjunction $F_1 \wedge \ldots \wedge F_n$ is valid or satisfiable or inconsistent or**
- •**a formula G follows from (axioms) $F_1 \wedge \ldots \wedge F_n$, denoted $F_1 \wedge \ldots \wedge F_n \models G$**

•**decidable** – there are algorithms that can solve the above problems (and always terminate)

•**NP-complete, but relatively efficient algorithms known (for propositional logic)**



Overview

•Propositional Logic

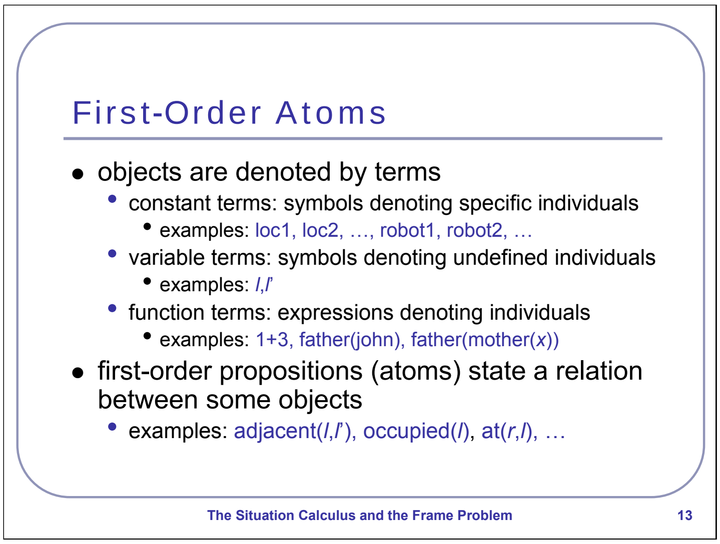
➧First-Order Predicate Logic

- ➧now: a more complex logic (sufficient for situation calculus)

•Representing States and Actions

•The Frame Problem

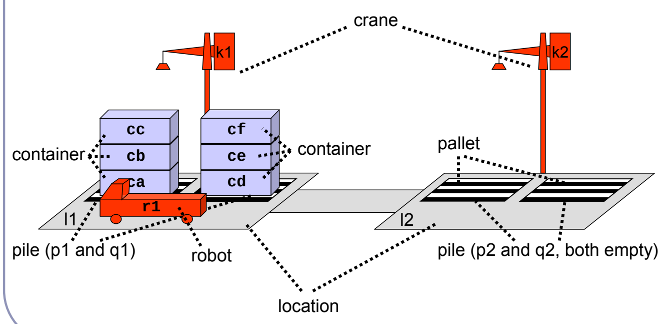
•Solving the Frame Problem

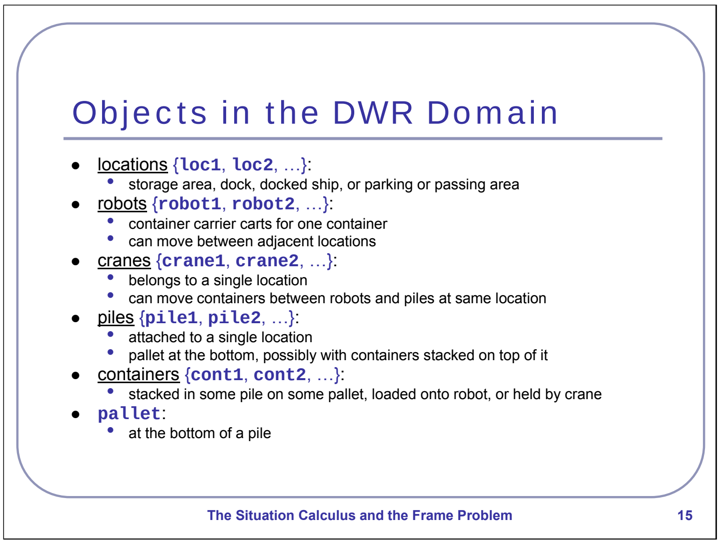


First-Order Atoms – like propositions but with internal structure

- **objects are denoted by terms**
  - **constant terms: symbols denoting specific individuals** or concepts (intangible objects)
    - **examples: loc1, loc2, …, robot1, robot2, …**
  - **variable terms: symbols denoting undefined individuals** usually bound by quantifiers
    - **examples: *l*,*l'***
  - **function terms: expressions denoting individuals** without introducing individual names: infinite domains!
    - **examples: 1+3, father(john), father(mother(*x*))**
- **first-order propositions (atoms) state a relation between some objects**
  - **examples: adjacent(*l*,*l'*), occupied(*l*), at(*r*,*l*), …**

DWR Example State





Objects in the DWR Domain (1)

•<u>locations</u> {loc1, loc2, …}:

- **storage area, dock, docked ship, or parking or passing area**
- do not necessarily have piles, e.g. parking or passing areas

•<u>robots</u> {robot1, robot2, …}:

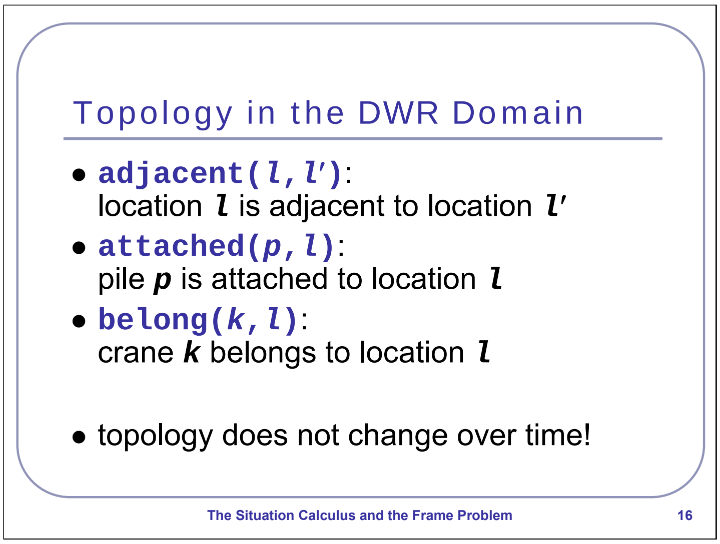
- **container carrier carts for one container**
- **can move between adjacent locations**
- can be loaded/unloaded by cranes at the same location
- at most one robot at one location at any one time

•<u>cranes</u> {crane1, crane2, …}:

- **belongs to a single location**
- **can move containers between robots and piles at same location**
- can load/unload containers onto/from robots, or take/put containers from/onto top of piles, all at the same location
- possibly multiple cranes per location

•<u>piles</u> {pile1, pile2, …}:

- **attached to a single location**
- locations with piles must also have cranes
- **pallet at the bottom, possibly with containers stacked on top of it**
- zero or more (unlimited number of) containers in a pile



Topology in the DWR Domain

•adjacent(*l*,*l'*): location *l* is adjacent to location *l'*

- •robots can move between adjacent locations

•attached(*p*,*l*): pile *p* is attached to location *l*

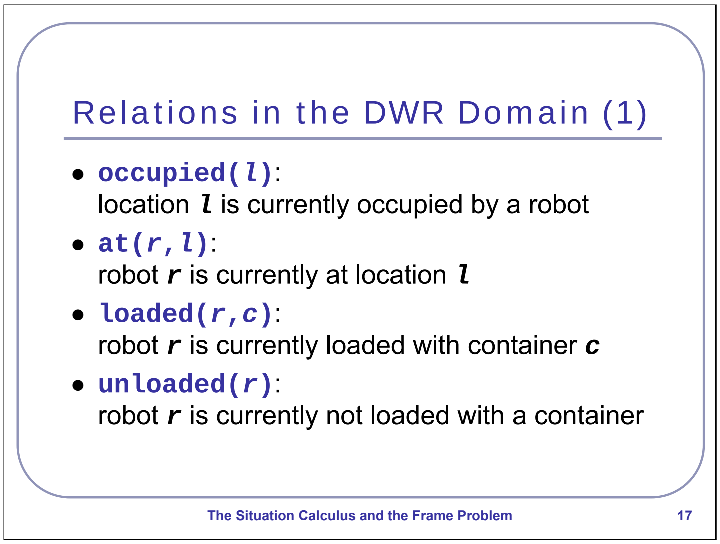
- •each pile is at exactly one location

•belong(*k*,*l*): crane *k* belongs to location *l*

- •cranes only have access to piles/robot at same location

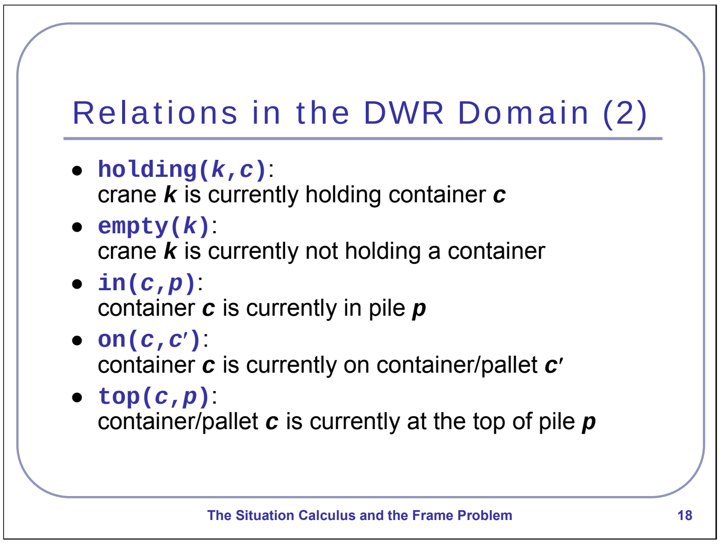
•topology does not change over time!

- •predicates denote fixed relationships (as opposed to fluents)



Relations in the DWR Domain (1)

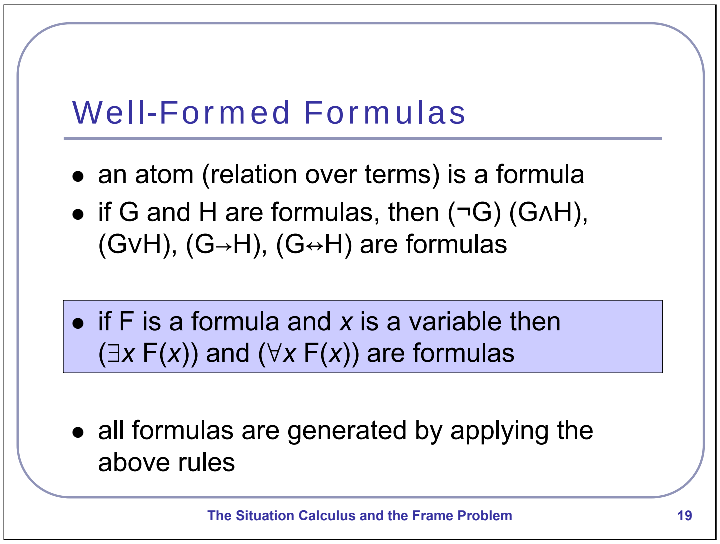
- **occupied(*l*): location *l* is currently occupied by a robot**
- **at(*r*,*l*): robot *r* is currently at location *l***
  - note: at(*r*,*l*) implies occupied(*l*)
- **loaded(*r*,*c*): robot *r* is currently loaded with container *c***
- **unloaded(*r*): robot *r* is currently not loaded with a container**
  - note: loaded(*r*,*c*) implies not unloaded(*r*)



Relations in the DWR Domain (2)

- **holding(*k*,*c*): crane *k* is currently holding container *c***
- **empty(*k*): crane *k* is currently not holding a container**
  - note: holding(*k*,*c*) implies not empty(*k*)
- **in(*c*,*p*): container *c* is currently in pile *p***
- **on(*c*,*c*’): container *c* is currently on container/pallet *c*’**
  - note: *c*’ may be a container or the pallet
- **top(*c*,*p*): container/pallet *c* is currently at the top of pile *p***
  - note: *c* may be a container or the pallet if there are no containers in the pile

- note: fluents are not independent!



Well-Formed Formulas

•an atom (relation over terms) is a formula

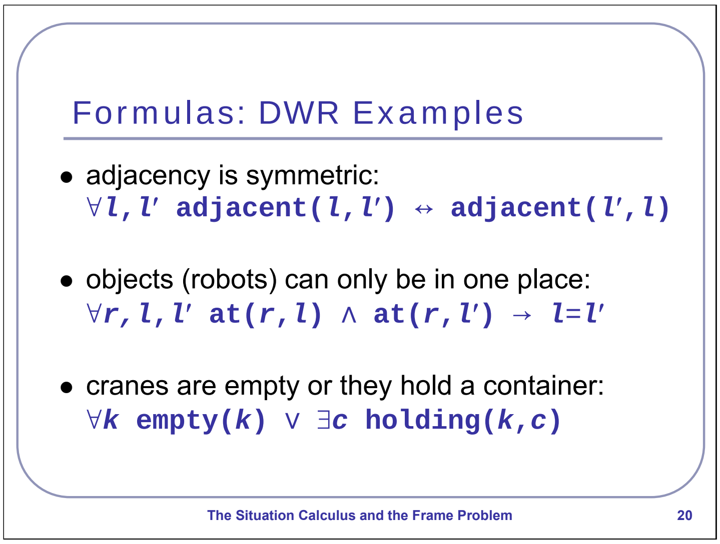
•if G and H are formulas, then $(\neg G)$ $(G \land H)$, $(G \lor H)$, $(G \rightarrow H)$, $(G \leftrightarrow H)$ are formulas

•new for first-order logic:

•if F is a formula and *x* is a variable then $(\exists x\ F(x))$ and $(\forall x\ F(x))$ are formulas

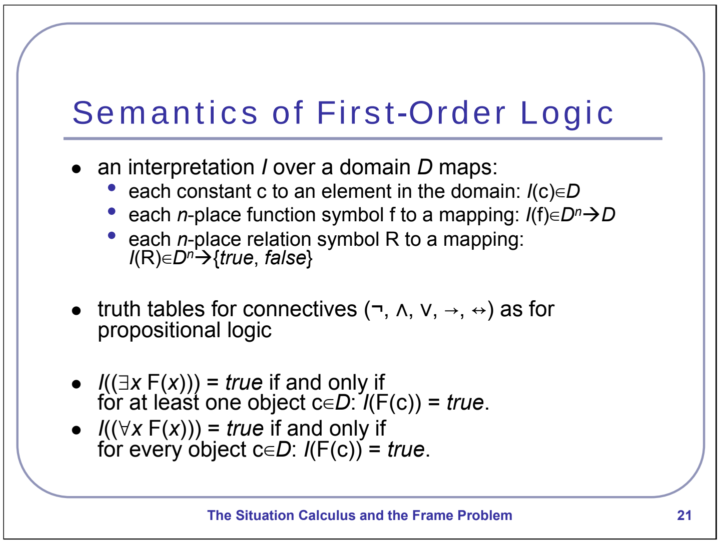
•existential and universal quantifiers over variable *x* and formula F containing variable *x*

•all formulas are generated by applying the above rules



Formulas: DWR Examples

- **adjacency is symmetric:** $\forall l, l'$ **adjacent**$(l, l') \leftrightarrow$ **adjacent**(l', l)
    - other possible properties of relations: reflexive, transitive
- **objects (robots) can only be in one place:** $\forall r, l, l'$ **at**$(r, l) \wedge$ **at**$(r, l') \rightarrow l = l'$
    - special relation: equality (assumed to be defined)
- **cranes are empty or they hold a container:** $\forall k$ **empty**$(k) \vee \exists c$ **holding**(k, c)



Semantics of First-Order Logic

•**an interpretation *I* over a domain *D* maps:** domain is just a set

- •**each constant c to an element in the domain: $I(c) \in D$**
- •**each *n*-place function symbol f to a mapping: $I(f) \in D^n \rightarrow D$**
- •**each *n*-place relation symbol R to a mapping: $I(R) \in D^n \rightarrow \{true, false\}$**
- •so far: interpretation assigns truth values to atoms

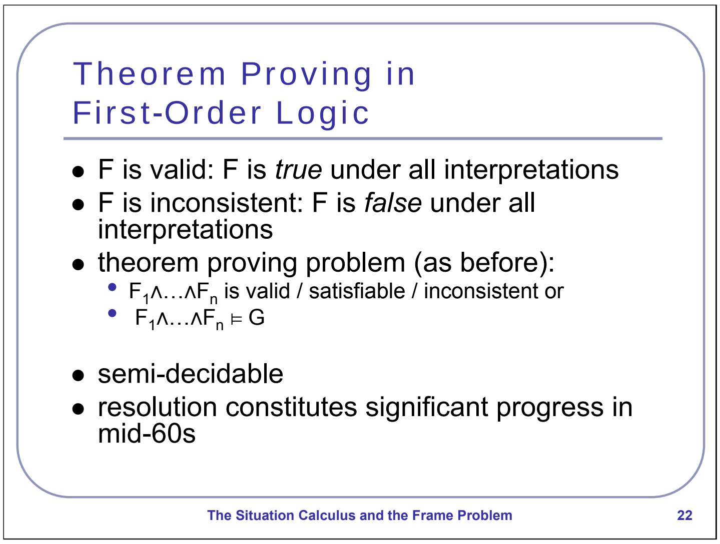
•**truth tables for connectives (¬, ∧, ∨, →, ↔) as for propositional logic**

•**$I((\exists x\, F(x))) = true$ if and only if for at least one object $c \in D$: $I(F(c)) = true$.**

- •existential quantifier: true if there exists an object that satisfies the formula

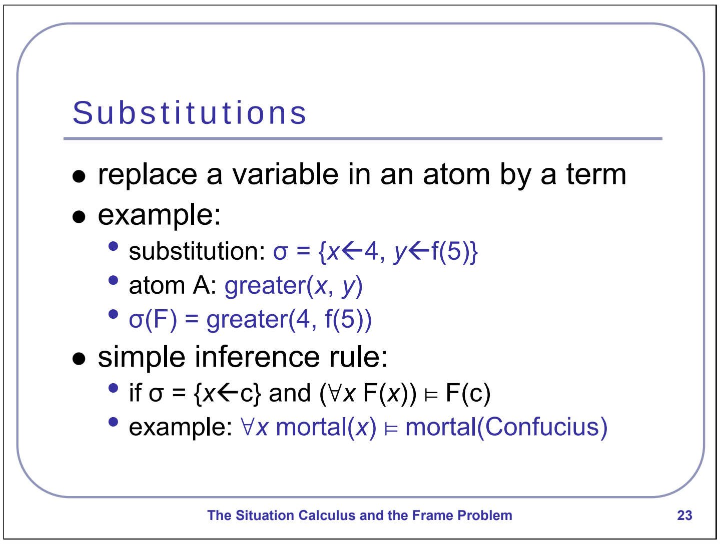
•**$I((\forall x\, F(x))) = true$ if and only if for every object $c \in D$: $I(F(c)) = true$.**

- •universal quantifier: true if every object satisfies the formula



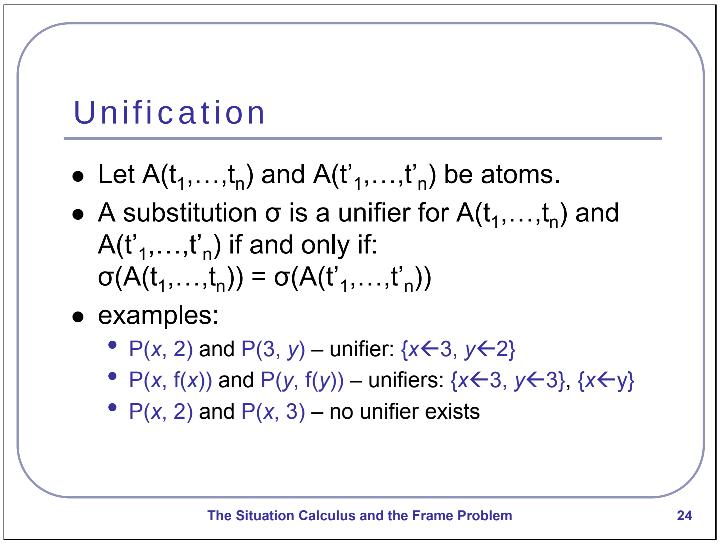
Theorem Proving in First-Order Logic

- **F is valid: F is *true* under all interpretations**
- **F is inconsistent: F is *false* under all interpretations**
    - essentially same as propositional logic
- **theorem proving problem (as before):**
    - **$F_1 \wedge \ldots \wedge F_n$ is valid / satisfiable / inconsistent or**
    - **$F_1 \wedge \ldots \wedge F_n \models G$**
- **semi-decidable**: if F is inconsistent an algorithm can find a proof
- **resolution constitutes significant progress in mid-60s**
    - hence the idea: use theorem prover as planner



Substitutions

- **replace a variable in an atom by a term**
    - variable must be free; complexity: replacement term may contain (other) variable
- **example:**
    - **substitution: $\sigma = \{x \leftarrow 4, y \leftarrow f(5)\}$**
    - **atom A: greater(x, y)**
    - **σ(F) = greater(4, f(5))**
- **simple inference rule:** instantiation
    - **if $\sigma = \{x \leftarrow c\}$ and $(\forall x\, F(x)) \models F(c)$**
    - **example: $\forall x$ mortal(x) $\models$ mortal(Confucius)**



Unification

•Let $A(t_1,\ldots,t_n)$ and $A(t'_1,\ldots,t'_n)$ be atoms.

- •predicate/relation: A; terms $t_1,\ldots,t_n,t'_1,\ldots,t'_n$

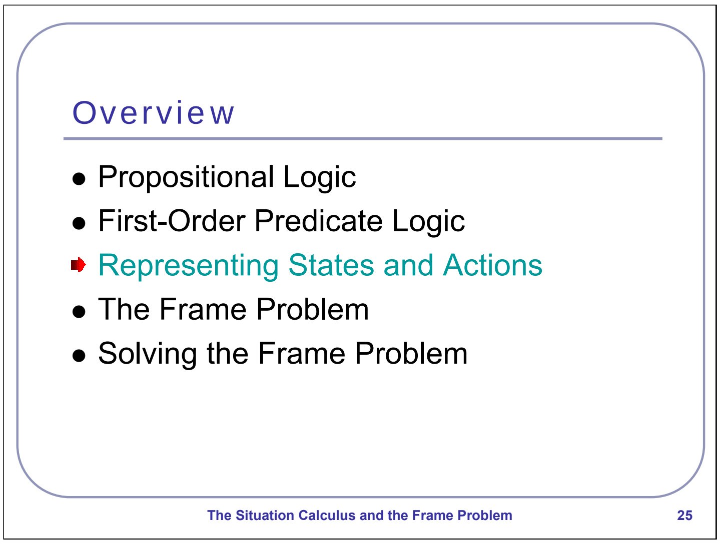
•A substitution σ is a unifier for $A(t_1,\ldots,t_n)$ and $A(t'_1,\ldots,t'_n)$ if and only if:
$\sigma(A(t_1,\ldots,t_n)) = \sigma(A(t'_1,\ldots,t'_n))$

- •replace variables such that atoms are equal

•examples:

- **•P(*x*, 2) and P(3, *y*) – unifier: {*x*←3, *y*←2}**
- **•P(*x*, f(*x*)) and P(*y*, f(*y*)) – unifiers: {*x*←3, *y*←3}, {*x*←y}**
  latter is more general
- **•P(*x*, 2) and P(*x*, 3) – no unifier exists**

•efficient algorithm for finding most general unifier is known



Overview

•Propositional Logic

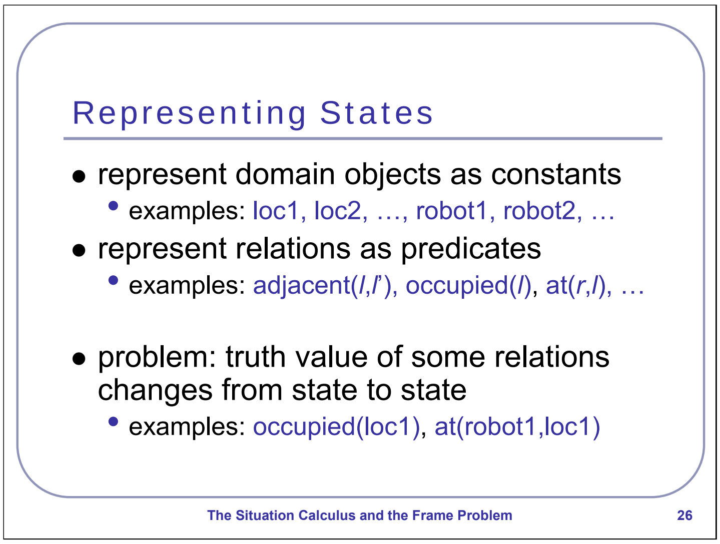
•First-Order Predicate Logic

•Representing States and Actions

- •now: an approach to representing and solving planning problems in first-order logic

•The Frame Problem

•Solving the Frame Problem



Representing States

•represent domain objects as constants

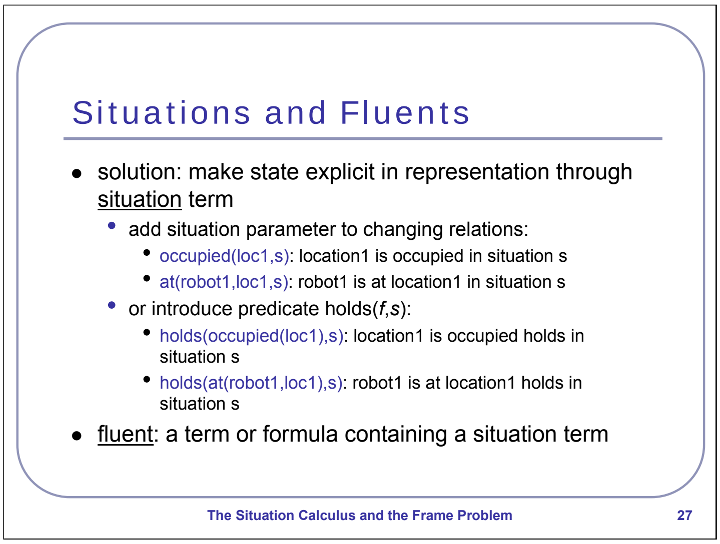
- **•examples: loc1, loc2, …, robot1, robot2, …**
- •represented by constant symbols

•represent relations as predicates

- **•examples: adjacent(*l*,*l’*), occupied(*l*), at(*r*,*l*), …**

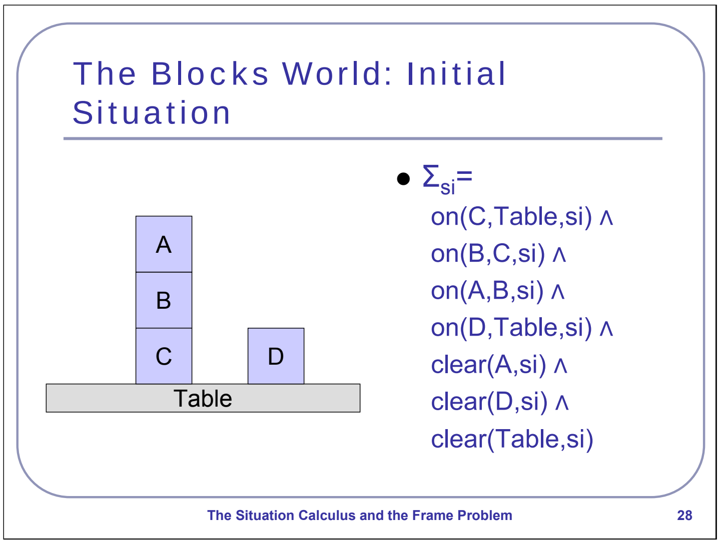
•problem: truth value of some relations changes from state to state

- •each state corresponds to a different logical theory
- **•examples: occupied(loc1), at(robot1,loc1)**
- •application of actions changes the truth values



Situations and Fluents

- **solution: make state explicit in representation through <u>situation</u> term**
    - sentences in FOPL are usually assumed to implicitly refer to the same state
    - situation term allows the naming of a state in which a relation may hold
    - **add situation parameter to changing relations:**
        - **occupied(loc1,s): location1 is occupied in situation s**
        - **at(robot1,loc1,s): robot1 is at location1 in situation s**
    - **or introduce predicate holds(*f*,*s*):**
        - **holds(occupied(loc1),s): location1 is occupied holds in situation s**
        - **holds(at(robot1,loc1),s): robot1 is at location1 holds in situation s**
    - both approaches are equivalent, but second approach turns relation into function term
- **<u>fluent</u>: a term or formula containing a situation term**
    - truth value changes between situations
- note: relations that do not change do not need to be related to situations



The Blocks World: Initial Situation

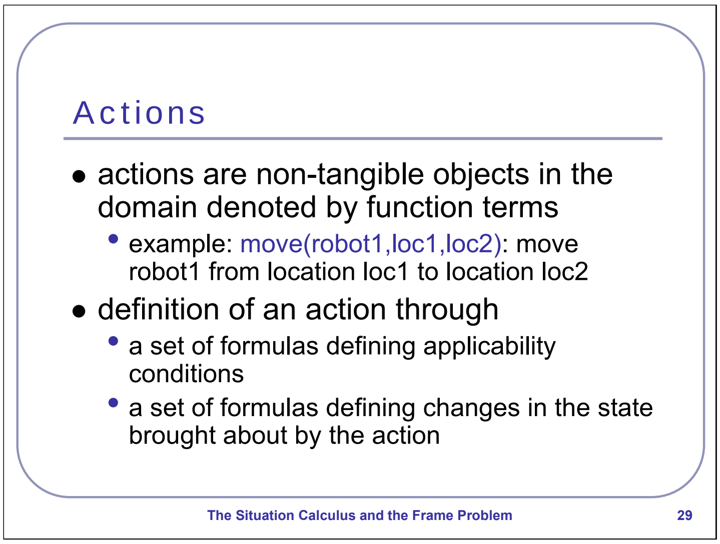
•[figure]

- •domain objects: blocks A, B, C, and D; the Table

•Σ_{si}=

- •si: the initial situation depicted here
- •fluents:
  - •on(*x,y,s*): denotes that block *x* is on block *y* in situation *s*
  - •clear(*x,s*): there is room on top of *x* for a block in *s*; *x* being a block or the Table which is always clear
- **•on(C,Table,si) ∧**
- **•on(B,C,si) ∧**
- **•on(A,B,si) ∧**
- **•on(D,Table,si) ∧**
- **•clear(A,si) ∧**
- **•clear(D,si) ∧**
- **•clear(Table,si)**

•note: cannot draw negative conclusions, as in ¬on(x,y,si) or ¬clear(x,si)



Actions

•actions are non-tangible objects in the domain denoted by function terms

> **•example: move(robot1,loc1,loc2): move robot1 from location loc1 to location loc2**
>
> •function symbol is action name or type, arguments are objects involved or manipulated

•definition of an action through

> **•a set of formulas defining applicability conditions**
>
> **•a set of formulas defining changes in the state brought about by the action**
>
> •actions are described in the same first-order language as states

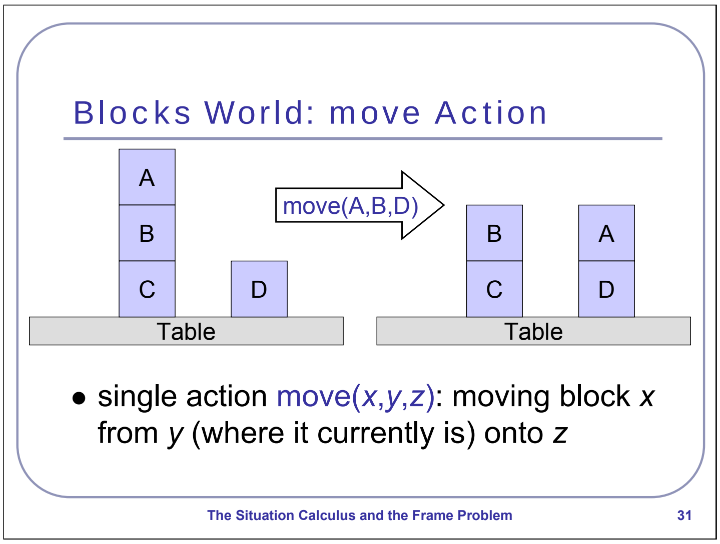
Blocks World: Applicability

- Δ_a=
  ∀*x*,*y*,*z*,*s*: applicable(move(*x*,*y*,*z*),*s*) ↔
  clear(*x*,*s*) ∧
  clear(*z*,*s*) ∧
  on(*x*,*y*,*s*) ∧
  x≠Table ∧
  x≠*z* ∧
  y≠*z*

Blocks World: Applicability

•Δ_a=

- •defined here for later reference to this formula
- •**∀*x*,*y*,*z*,*s*: applicable(move(*x*,*y*,*z*),*s*) ↔**
  - •**clear(*x*,*s*) ∧**
    - •the block to be moved must be clear
  - •**clear(*z*,*s*) ∧**
    - •the place where we move it must be clear
  - •**on(*x*,*y*,*s*) ∧**
    - •condition used to bind *y*
  - •***x*≠Table ∧**
    - •cannot move the table
  - •***x*≠*z* ∧**
    - •cannot move the block onto itself
  - •***y*≠*z***
    - •origin and destination should be different



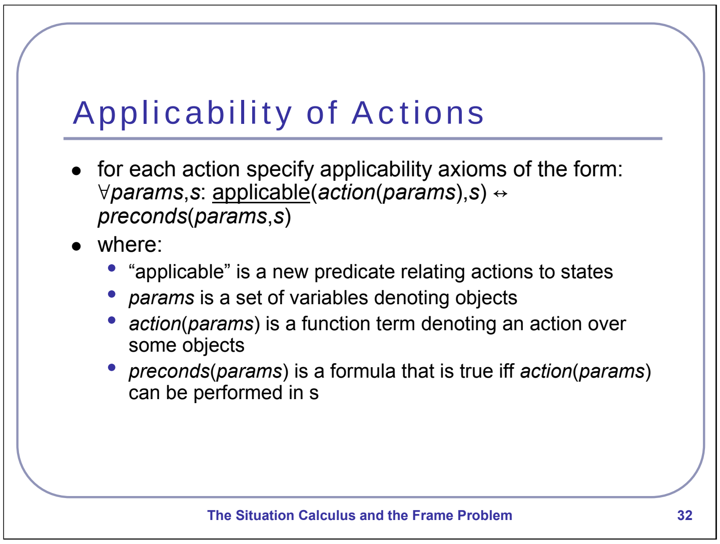
Blocks World: move Action

•[figure]

- •left: situation before the action is performed
- •action: move block A from block B onto block D
- •right: situation after the action has been performed

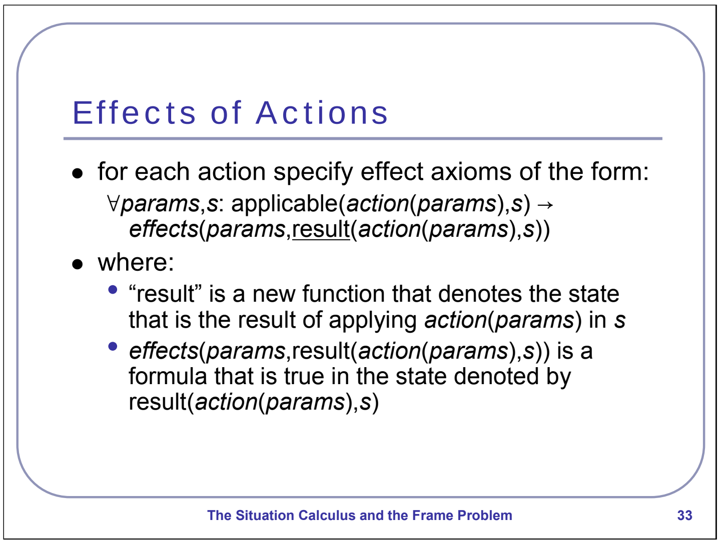
•single action move(*x*,*y*,*z*): moving block *x* from *y* (where it currently is) onto *z*

- •either *y* or *z* may be the Table, but not *x*



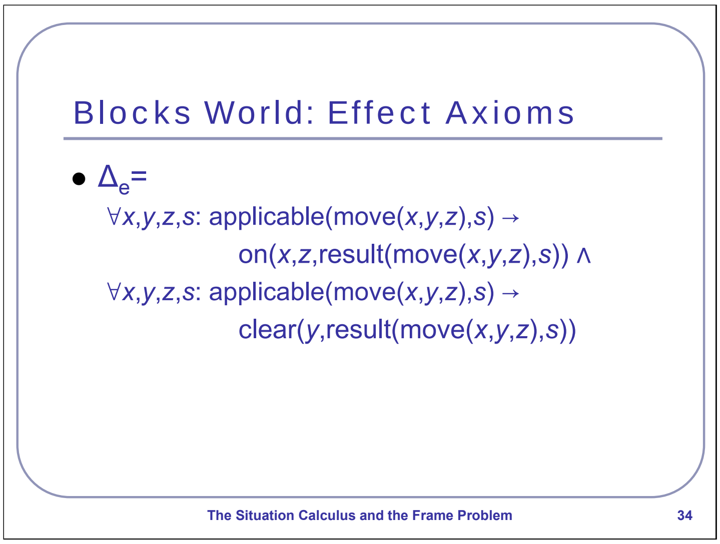
Applicability of Actions

- **for each action specify applicability axioms of the form:**
  - **$\forall$*params*,*s*: <u>applicable</u>(*action*(*params*),*s*) ↔ *preconds*(*params*,*s*)**
    - similar for version based on holds-predicate
- **where:**
  - **"applicable" is a new predicate relating actions to states**
    - true iff action is applicable in state
  - ***params* is a set of variables denoting objects**
    - the objects manipulated by the action in some way
  - ***action*(*params*) is a function term denoting an action over some objects**
  - ***preconds*(*params*) is a formula that is true iff *action*(*params*) can be performed in s**
    - can be any first-order formula involving quantifiers and connectives



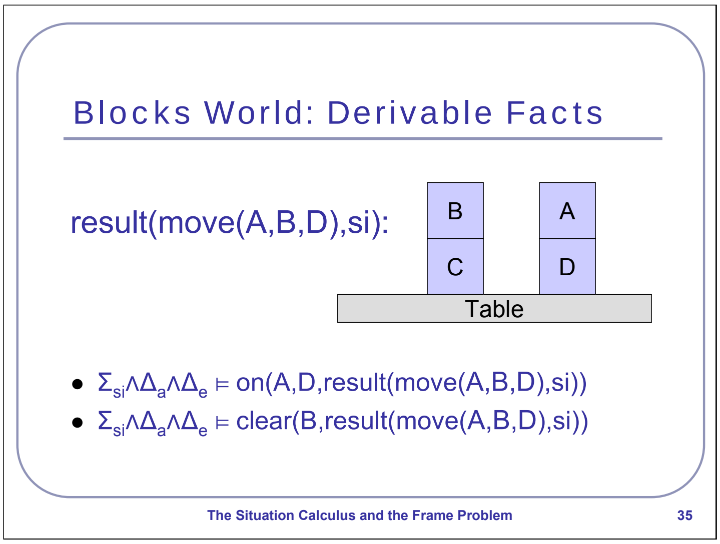
Effects of Actions

- **for each action specify effect axioms of the form:**
  - **$\forall$*params*,*s*: applicable(*action*(*params*),*s*) $\rightarrow$ *effects*(*params*,result(*action*(*params*),*s*))**
    - similar for version based on holds-predicate
- **where:**
  - **"result" is a new function that denotes the state that is the result of applying *action*(*params*) in *s***
    - function that maps an action and a situation into a situation
  - ***effects*(*params*,result(*action*(*params*),*s*)) is a formula that is true in the state denoted by result(*action*(*params*),*s*)**
    - can be any first-order formula involving quantifiers and connectives



Blocks World: Effect Axioms

- Δ_e=
  - **$\forall x,y,z,s$: applicable(move(x,y,z),s) →**
    - **on(x,z,result(move(x,y,z),s)) ∧**
      - x will now be on z
  - **$\forall x,y,z,s$: applicable(move(x,y,z),s) →**
    - **clear(y,result(move(x,y,z),s))**
      - y will be clear as a result of the move
- note: no negative effects specified, e.g. x is no longer on y



Blocks World: Derivable Facts

•**[figure]** shows the result of moving A from B onto D

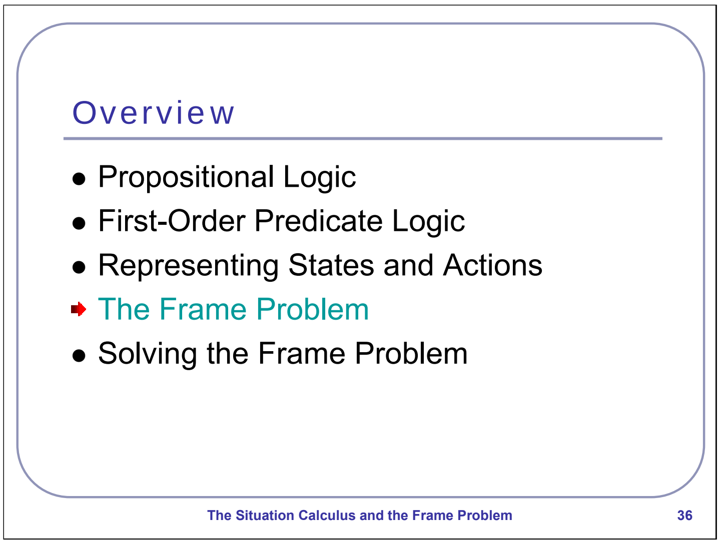
•$\Sigma_{si} \wedge \Delta_a \wedge \Delta_e \models$ on(A,D,**result(move(A,B,D),si))**

  •it follows that A is now on D and

•$\Sigma_{si} \wedge \Delta_a \wedge \Delta_e \models$ clear(B,**result(move(A,B,D),si))**

  •it follows that B is now clear

•these facts can be derived by any sound and complete theorem proving algorithm



Overview

•Propositional Logic

•First-Order Predicate Logic

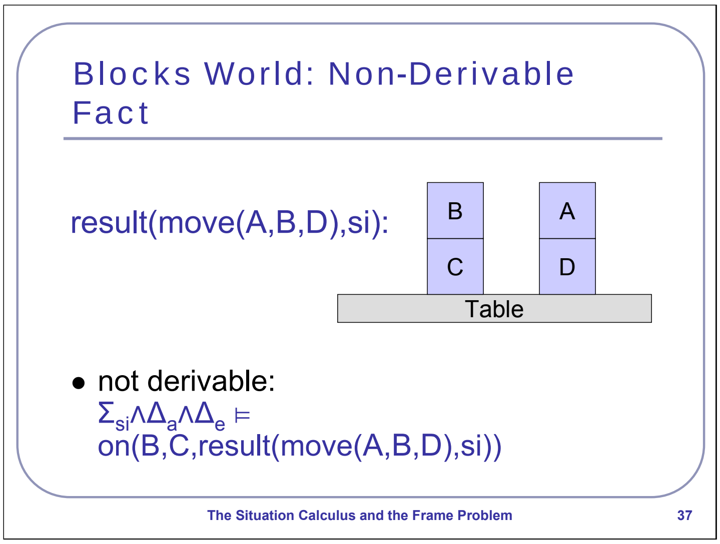
•Representing States and Actions

- •just done: an approach to representing and solving planning problems in first-order logic

•The Frame Problem

- •now: defining the frame problem

•Solving the Frame Problem

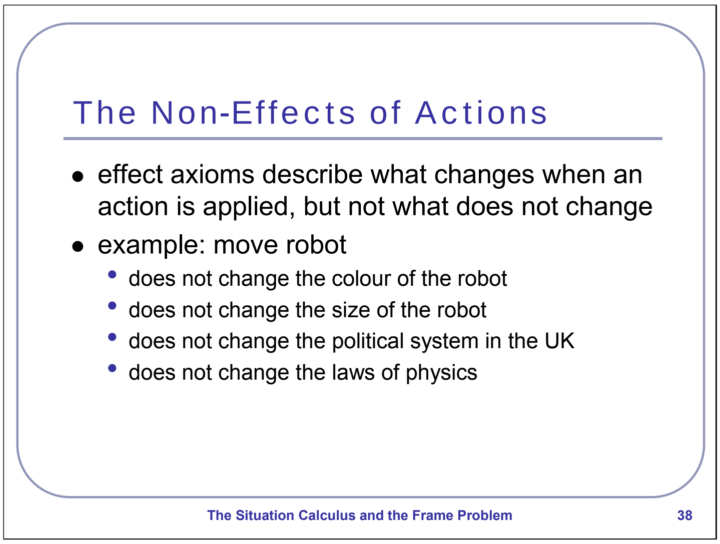


Blocks World: Non-Derivable Fact

•**[figure]** as before

•not derivable: $\Sigma_{si} \wedge \Delta_a \wedge \Delta_e \vDash$ on(B,C,**result(move(A,B,D),si))**

- •the fact that B is still on C does not logically follow from the theory
- •effect axioms list only what is true as a direct result of an action, not what stays true



The Non-Effects of Actions

•effect axioms describe what changes when an action is applied, but not what does not change

  •frame problem: need to explicitly describe what does not change when an action is performed

•example: move robot

  •does not change the colour of the robot

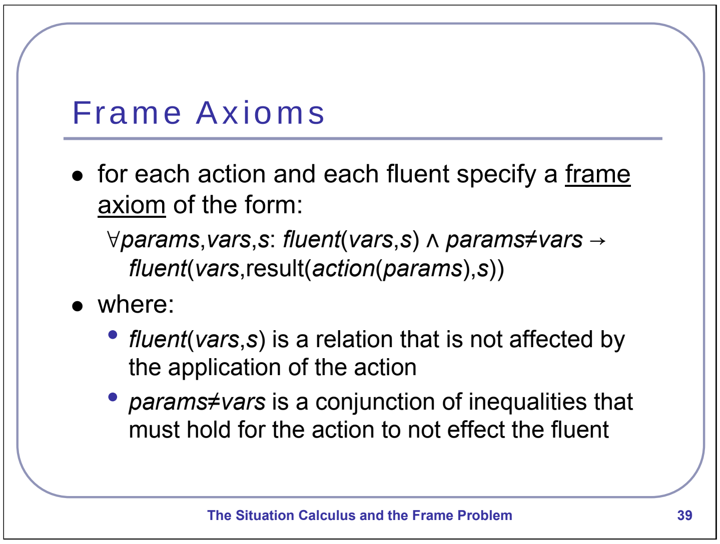
  •does not change the size of the robot

  •does not change the political system in the UK

  •does not change the laws of physics

•there is an infinite number of facts that do not change

•but also: butterfly effect – everything affects everything



Frame Axioms

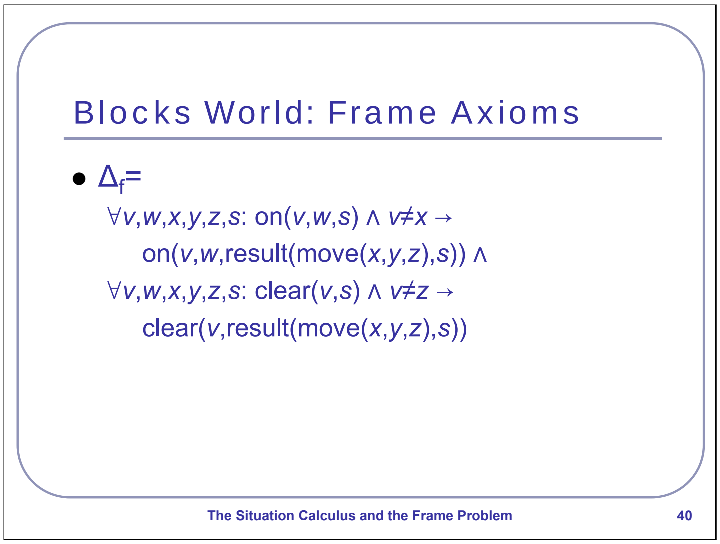
- frame axioms capture persistence of fluents that are unaffected by actions

•for each action and each fluent specify a <u>frame axiom</u> of the form:

- **$\forall \mathit{params},\mathit{vars},s$: $\mathit{fluent}(\mathit{vars},s) \wedge \mathit{params} \neq \mathit{vars} \rightarrow \mathit{fluent}(\mathit{vars},\mathrm{result}(\mathit{action}(\mathit{params}),s))$**
  - inequality needed if fluent is unaffected depending on parameters

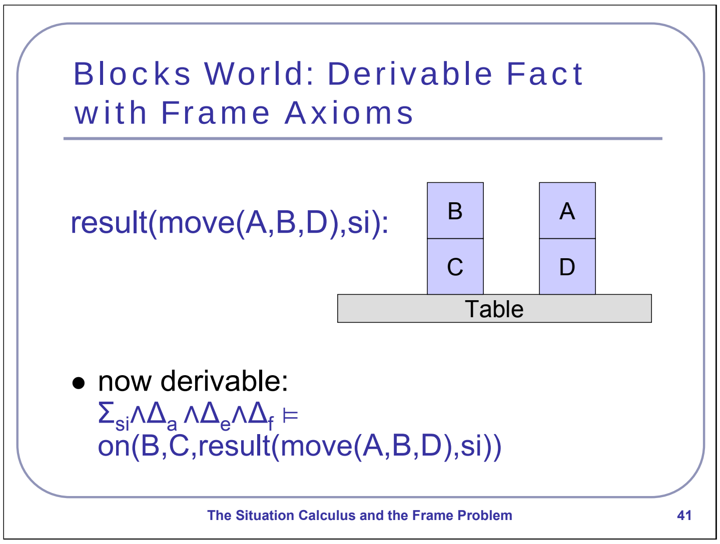
•where:

- ***fluent*(*vars*,*s*) is a relation that is not affected by the application of the action**
  - generally, *vars* are different from *params*
- ***params≠vars* is a conjunction of inequalities that must hold for the action to not effect the fluent**
  - see examples that follow



Blocks World: Frame Axioms

- Δ_f=
  - $\forall v,w,x,y,z,s$: **on(v,w,s) $\wedge$ $v \neq x \rightarrow$**
    - **on(v,w,result(move(x,y,z),s)) $\wedge$**
      - if v is not the block that is being moved (inequality) then "v on w" persists
  - $\forall v,w,x,y,z,s$: **clear(v,s) $\wedge$ $v \neq z \rightarrow$**
    - **clear(v,result(move(x,y,z),s))**
      - if v is not the place the block x is moved onto (inequality) then "v is clear" persists



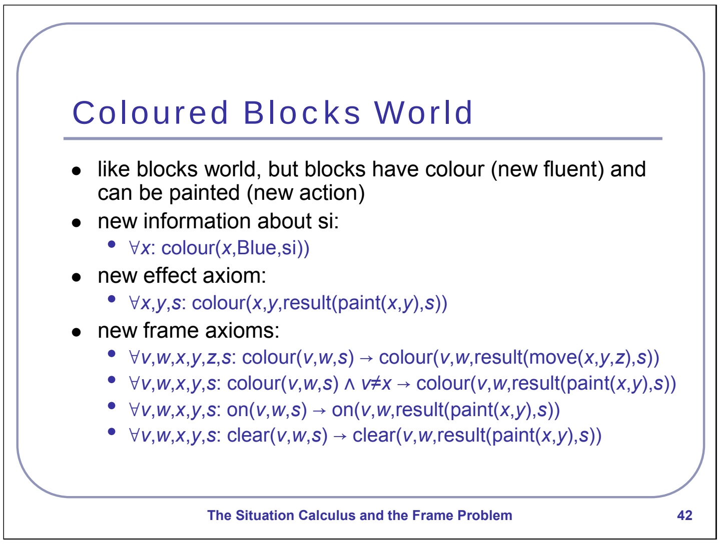
Blocks World: Derivable Fact with Frame Axioms

•**[figure]** as before

•now derivable: $\Sigma_{si} \wedge \Delta_a \wedge \Delta_e \wedge \Delta_f \models$ on(B,C,**result(move(A,B,D),si))**

- •fact that B remains on C can now be proven

•need for two frame axioms might be surprising but gives desired result



Coloured Blocks World

•like blocks world, but blocks have colour (new fluent) and can be painted (new action)

- •colour(x,y) denotes that block x has colour y
- •paint(x,y) denotes the action of painting block x in colour y (no applicability conditions)

•new information about si:

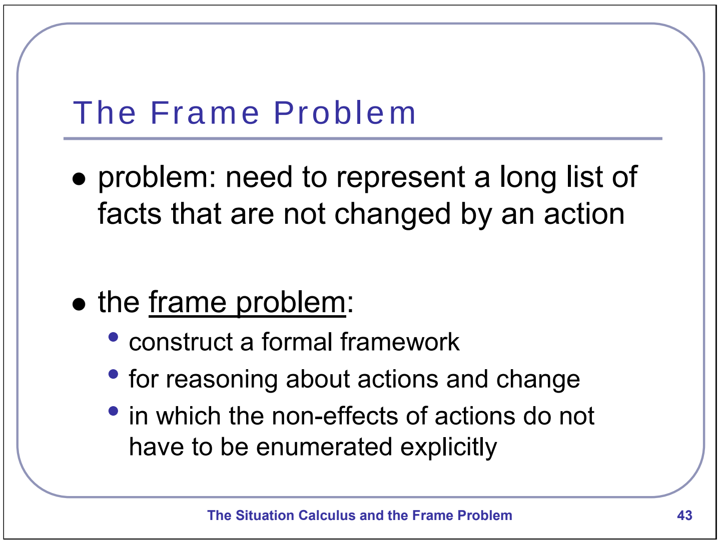
- •$\forall x: \mathrm{colour}(x,\mathrm{Blue},\mathrm{si}))$

•new effect axiom:

- •$\forall x,y,s: \mathrm{colour}(x,y,\mathrm{result}(\mathrm{paint}(x,y),s))$

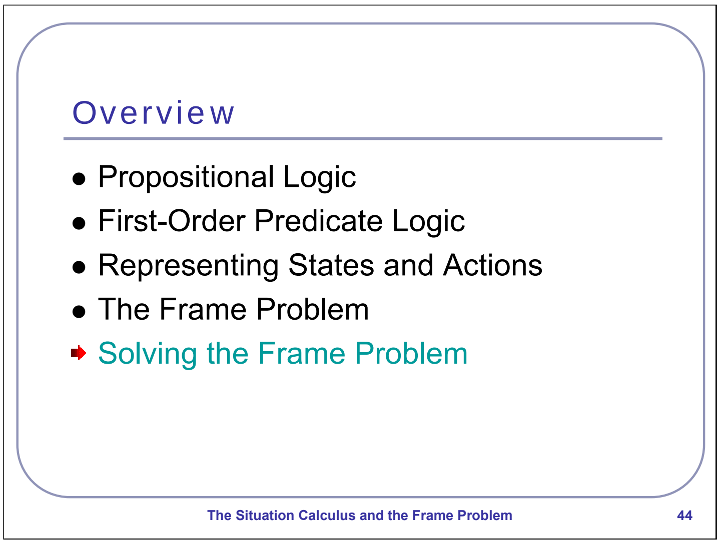
•new frame axioms:

- •$\forall v,w,x,y,z,s: \mathrm{colour}(v,w,s) \rightarrow \mathrm{colour}(v,w,\mathrm{result}(\mathrm{move}(x,y,z),s))$
    - •moving a block does not change the colour of any block
- •$\forall v,w,x,y,s: \mathrm{colour}(v,w,s) \land v \neq x \rightarrow \mathrm{colour}(v,w,\mathrm{result}(\mathrm{paint}(x,y),s))$
    - •painting a block does not change the colour of any other block
- •$\forall v,w,x,y,s: \mathrm{on}(v,w,s) \rightarrow \mathrm{on}(v,w,\mathrm{result}(\mathrm{paint}(x,y),s))$
    - •painting a block does not change which block is on which
- •$\forall v,w,x,y,s: \mathrm{clear}(v,w,s) \rightarrow \mathrm{clear}(v,w,\mathrm{result}(\mathrm{paint}(x,y),s))$
    - •painting a block does not change which blocks a re clear



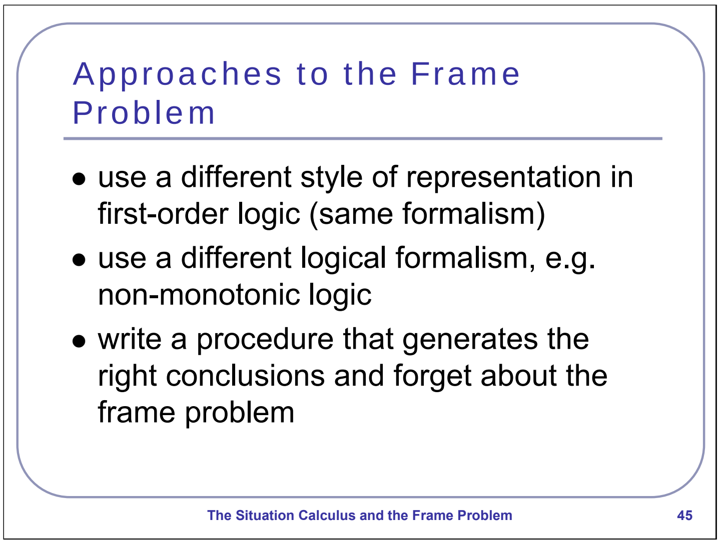
The Frame Problem

- **problem: need to represent a long list of facts that are not changed by an action**
  - description of what does not change is considerably larger than of what does change: number of frame axioms is number of actions times number of fluents
    - add a new fluent: add (number of actions) new frame axioms
    - add a new action: add (number of fluents) new frame axioms
  - frame problem first described by McCarthy and Hayes (1969):
- **the <u>frame problem</u>:**
  - **construct a formal framework**
  - **for reasoning about actions and change**
  - **in which the non-effects of actions do not have to be enumerated explicitly**
    - what does not change is felt to be common sense; there should be no need to write it down explicitly



Overview

- **Propositional Logic**
- **First-Order Predicate Logic**
- **Representing States and Actions**
- **The Frame Problem**
  - just done: defining the frame problem
- **Solving the Frame Problem**
  - now: types of approaches to the frame problem



Approaches to the Frame Problem

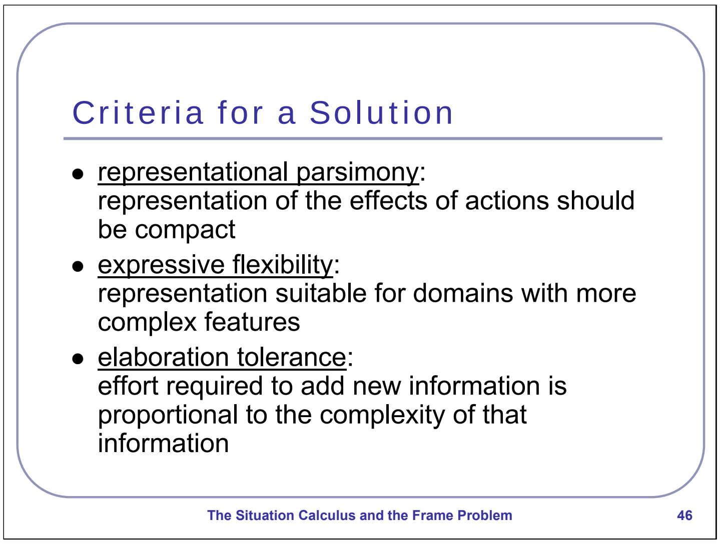
•use a different style of representation in first-order logic (same formalism)

- •various have been tried but the frame problem keeps showing up

•use a different logical formalism, e.g. non-monotonic logic

•write a procedure that generates the right conclusions and forget about the frame problem

- •the STRIPS approach: is it a representation?
- •logical vs. computational aspect of the frame problem
- •rest of this course follows mostly this approach



Criteria for a Solution

•representational parsimony: representation of the effects of actions should be compact

- •size of the representation should be roughly proportional to the complexity of the domain (number of actions + number of fluents)
- •not true for situation calculus (so far)

•expressive flexibility: representation suitable for domains with more complex features

- •complex features: ramifications (e.g. three blocks on top of each other form a stack), concurrent actions, non-deterministic actions, continuous change

•elaboration tolerance: effort required to add new information is proportional to the complexity of that information

- •ideally, new action or fluent should be added (appended) to existing theory and not require a complete reconstruction

The Universal Frame Axiom

- frame axiom for all actions, fluents, and situations: $\forall a,f,s$: holds(f,s) ∧ ¬<u>affects</u>(a,f,s) → holds(f,result(a,s))
- where “affects” is a new predicate that relates actions, fluents, and situations
- ¬affects(a,f,s) is true if and only if the action a does not change the value of the fluent f in situation s

The Universal Frame Axiom

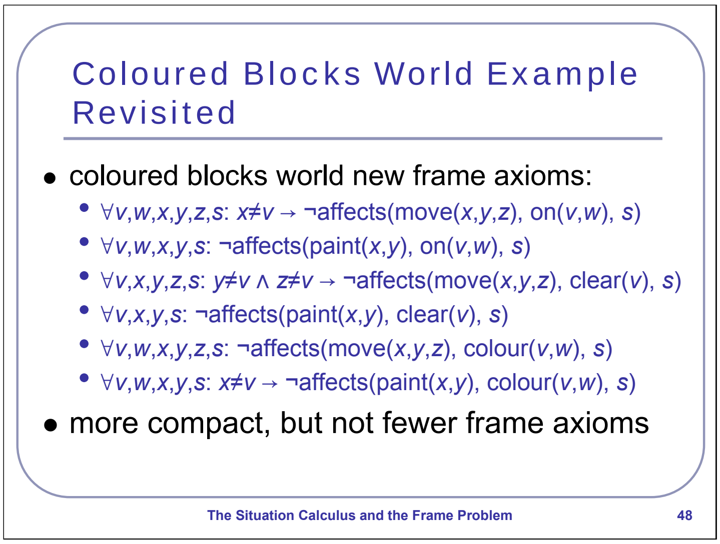
•approach: different style of representation in first-order logic

•frame axiom for all actions, fluents, and situations: $\forall a,f,s$: holds(f,s) ∧ ¬<u>affects</u>(a,f,s) → holds(f,result(a,s))

•requires different style of representation with fluent as function term

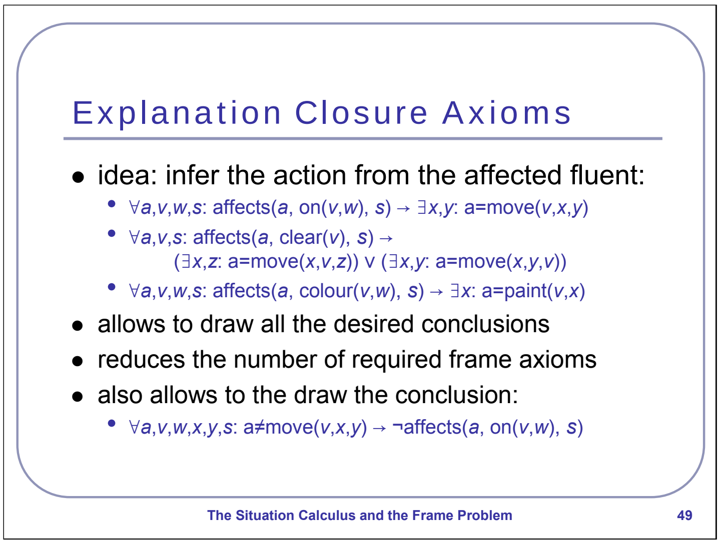
•where “affects” is a new predicate that relates actions, fluents, and situations

•¬affects(a,f,s) is true if and only if the action a does not change the value of the fluent f in situation s



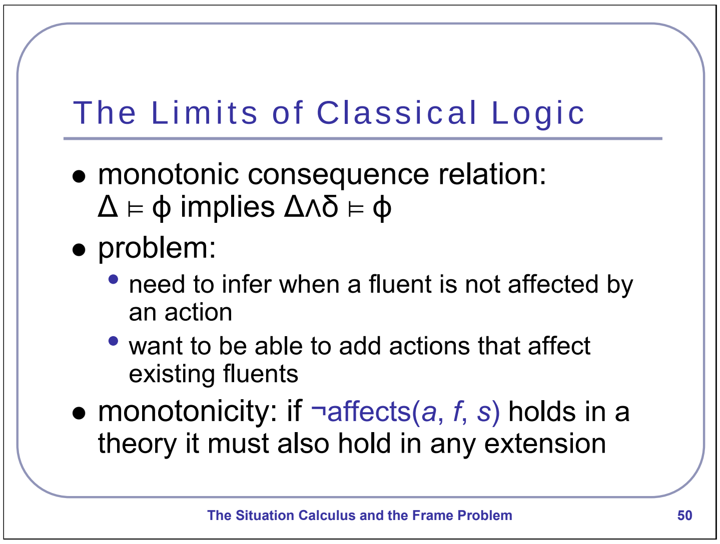
Coloured Blocks World Example Revisited

- **coloured blocks world new frame axioms:**
  - **$\forall v,w,x,y,z,s$: $x \neq v \rightarrow \neg$affects(move($x,y,z$), on($v,w$), s)**
  - **$\forall v,w,x,y,s$: $\neg$affects(paint(x,y), on(v,w), s)**
  - **$\forall v,x,y,z,s$: $z \neq v \rightarrow \neg$affects(move($x,y,z$), clear($v$), s)**
  - **$\forall v,x,y,s$: $\neg$affects(paint(x,y), clear(v), s)**
  - **$\forall v,w,x,y,z,s$: $\neg$affects(move(x,y,z), colour(v,w), s)**
  - **$\forall v,w,x,y,s$: $x \neq v \rightarrow \neg$affects(paint($x,y$), colour($v,w$), s)**
    - gives exactly the same conclusions as previous representation
- **more compact, but not fewer frame axioms**
  - still (number of actions) times (number of fluents) frame axioms required



Explanation Closure Axioms

- **idea: infer the action from the affected fluent:**
  - **∀*a*,*v*,*w*,*x*,*y*,*z*,*s*: affects(*a*, on(*v*,*w*), *s*) → a=move(*x*,*y*,*z*)**
  - **∀*a*,*v*,*x*,*y*,*z*,*s*: affects(*a*, clear(*v*), *s*) → a=move(*x*,*y*,*z*)**
  - **∀*a*,*v*,*w*,*x*,*y*,*s*: affects(*a*, colour(*v*,*w*), *s*) → a=paint(*x*,*y*)**
- **allows to draw all the desired conclusions**
- **reduces the number of required frame axioms**
- **also allows to the draw the conclusion:**
  - **∀*a*,*v*,*w*,*x*,*y*,*z*,*s*: a≠move(*x*,*y*,*z*) → ¬affects(*a*, on(*v*,*w*), *s*)**
- representational parsimony: yes; expressive flexibility: ?; elaboration tolerance: no



The Limits of Classical Logic

- **monotonic consequence relation: $\Delta \models \phi$ implies $\Delta \wedge \delta \models \phi$**
  - adding a formula does not invalidate previous conclusions
- **problem:**
  - **need to infer when a fluent is not affected by an action**
    - need to infer the necessary condition under which the fluent is affected
  - **want to be able to add actions that affect existing fluents**
    - want to admit the possibility of new actions or effects
    - elaboration tolerance: add these without modifying the existing theory
    - hence: previous consequences still valid in extended theory
- **monotonicity: if ¬affects(*a*, *f*, *s*) holds in a theory it must also hold in any extension**
  - hence: no new action can affect a pre-existing fluent

Using Non-Monotonic Logics

- non-monotonic logics rely on default reasoning:
  - jumping to conclusions in the absence of information to the contrary
  - conclusions are assumed to be true by default
  - additional information may invalidate them
- application to frame problem:
  - explanation closure axioms are default knowledge
  - effect axioms are certain knowledge

Using Non-Monotonic Logics

•non-monotonic logics rely on default reasoning:

•jumping to conclusions in the absence of information to the contrary

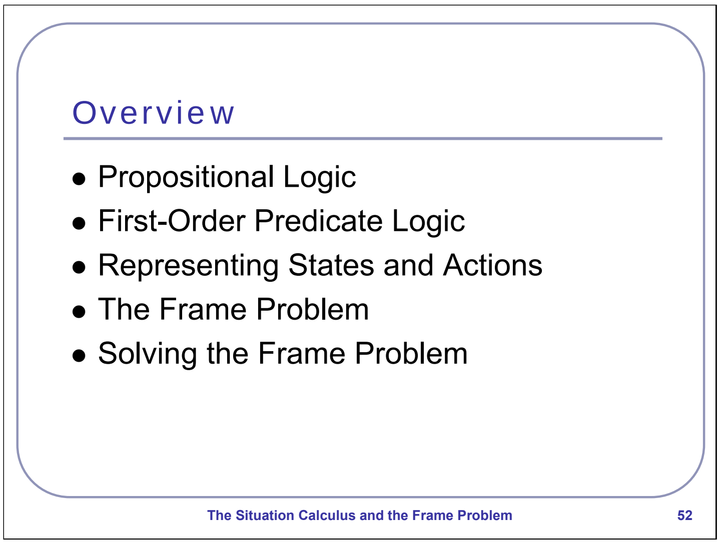
•conclusions are assumed to be true by default

•additional information may invalidate them

•application to frame problem:

•explanation closure axioms are default knowledge

•effect axioms are certain knowledge



Overview

- **Propositional Logic**
- **First-Order Predicate Logic**
- **Representing States and Actions**
- **The Frame Problem**
- **Solving the Frame Problem**
    - just done: types of approaches to the frame problem