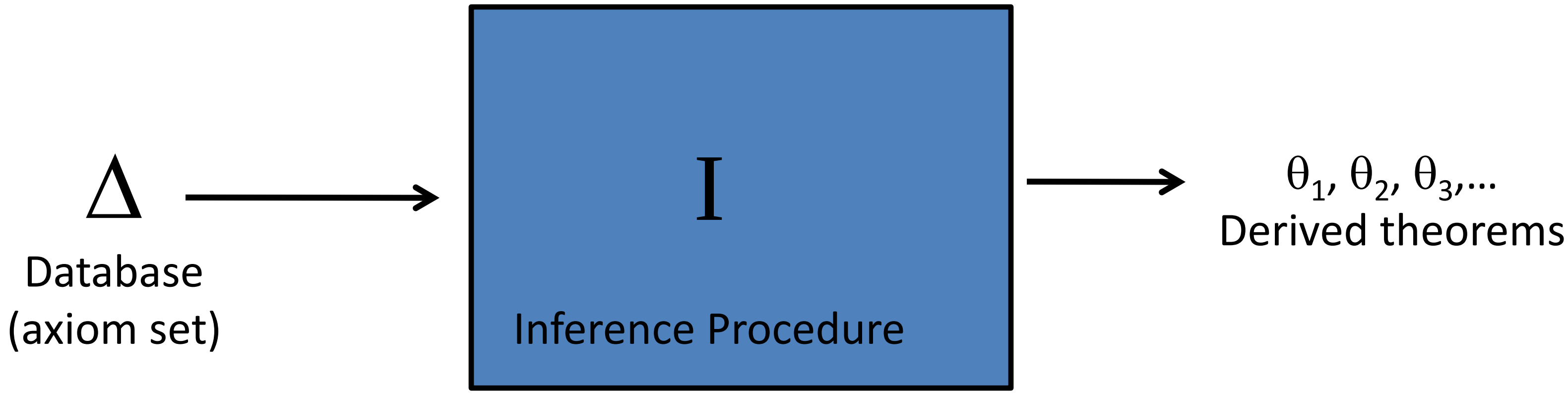
- Homework 2A due today
- Homework 2B due Tuesday (9/28)

More on Soundness and Completeness of inference procedures

Derivation requires an inference procedure
Entailment does NOT



Inference Procedure:

- choose one or more inference rules
- choose conventional (derive the goal directly)
  - or refutational (add negated goal, derive contradiction)
- (if ambiguous we will assume conventional)

Intrinsic properties of I (soundness and completeness)
force important relations between Δ and $\{\theta_i\}$

Completeness

An inference procedure is *complete* iff for any database (axiom set), any sentence entailed by the database can be derived from the database using the inference procedure.

It may not be possible for an inference procedure to derive a sentence even though the sentence is entailed by the database. Such an inference procedure is *incomplete*.

Given a database, a complete inference procedure must derive everything entailed by it (including all tautologies).

Soundness

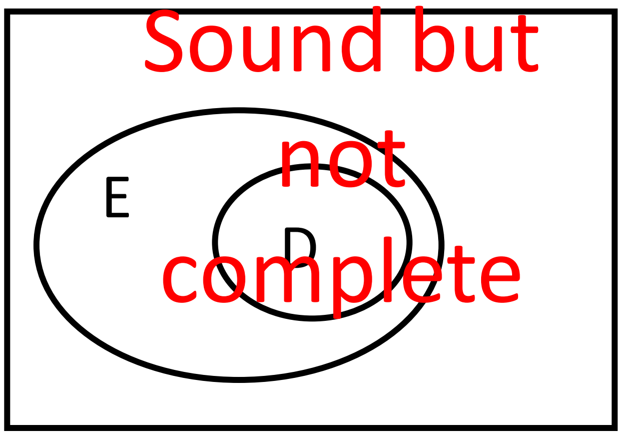
An inference procedure is *sound* iff for any database (axiom set) every sentence derivable from the database using the inference procedure is entailed by the database.

Some inference procedure may derive statements that do not follow logically from the database. Such an inference procedure is *unsound*.

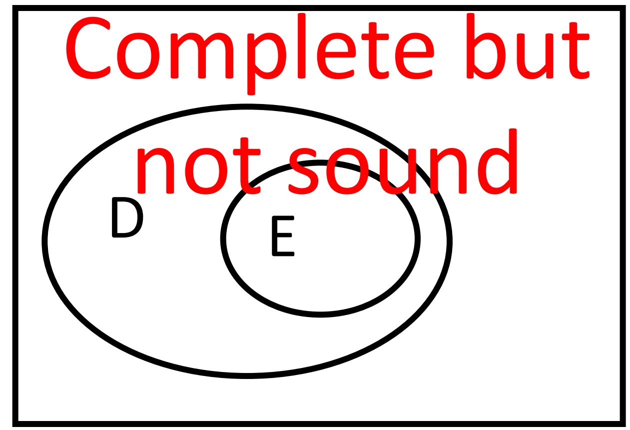
Given a database, a sound inference procedure must only derive statements that are entailed by it (that logically follow from it).

For all axiom sets we have these relationships

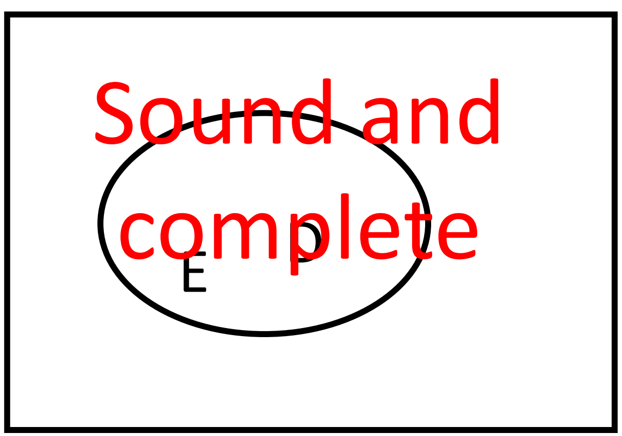
E: entailed WFFs D: derivable WFFs



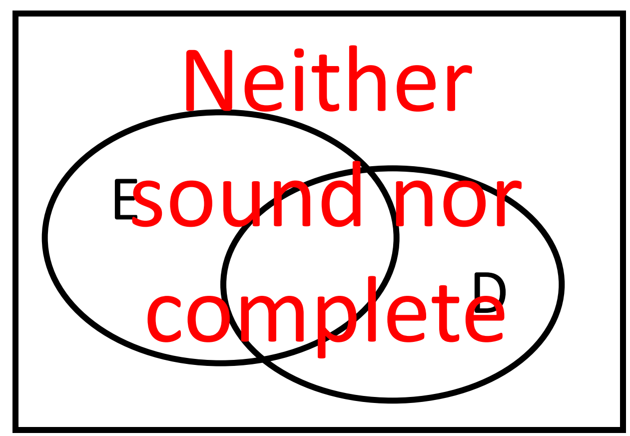
All WFFs



All WFFs



All WFFs



All WFFs

Modus Ponens

$$\frac{\Theta \Rightarrow \Psi \qquad \Theta}{\Psi}$$

Sound but not complete

Abduction

$$\frac{\Theta \Rightarrow \Psi \qquad \Psi}{\Theta}$$

Neither sound nor complete

Resolution

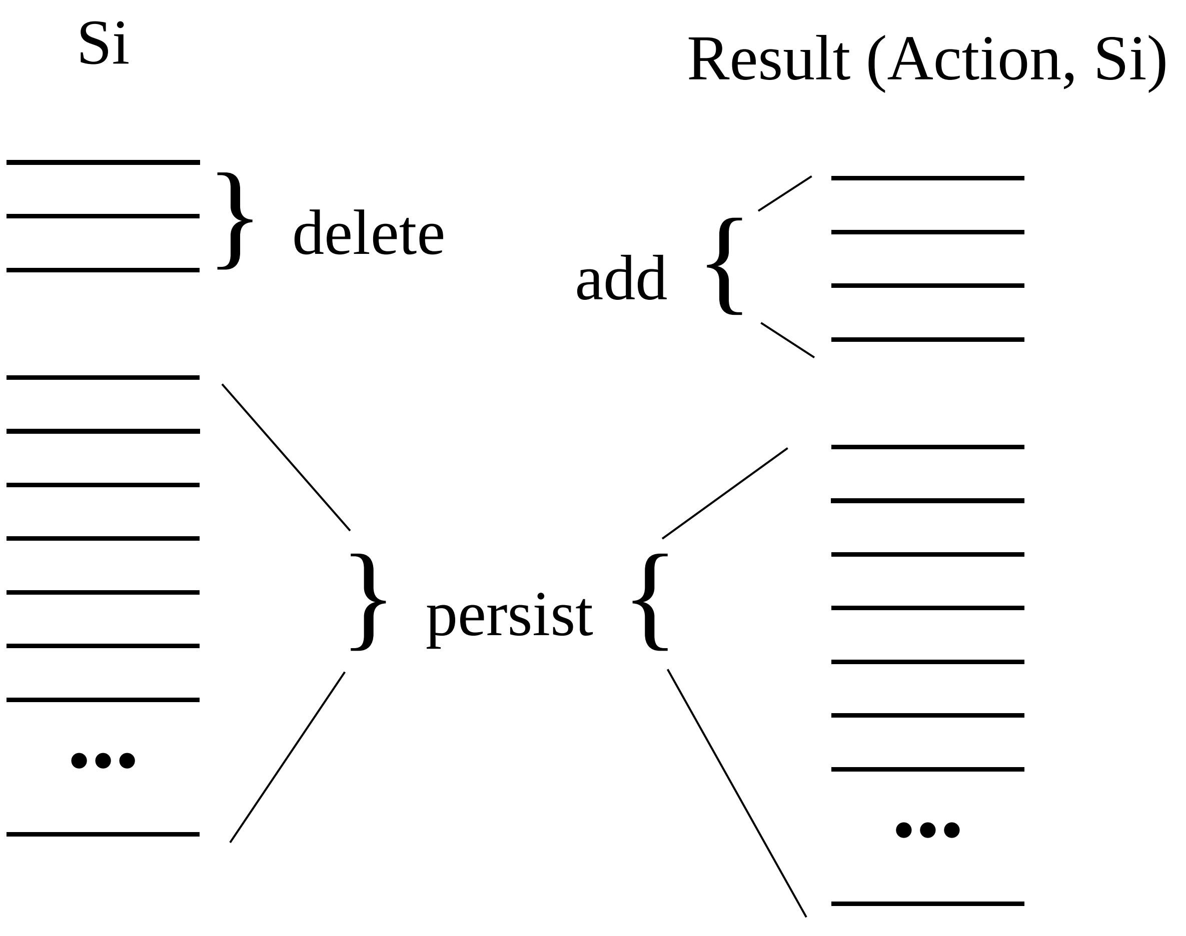
$$\frac{\begin{array}{c}\alpha \vee \beta \\ \neg\beta \vee \gamma\end{array}}{\alpha \vee \gamma}$$

Sound but not complete

Last time…
Situation Calculus vs Strips

World Changes

Action must fully define resulting world state



Operators

In Situation Calculus

Specify fluents
- Add set
- Persist set

No mention =
- no inference path

By default
- fluents are Deleted

In Strips

Specify fluents
- Delete set
- Add set

By default
- fluents Persist

More concise because usually
|Persist| >> |Delete|

Strips Operators

- Preconditions - list of positive literals
- Effects also positive literals (N.B. below)
  - Delete list - things to be retracted
  - Add list - things to be asserted
- Effects can be combined in one list (as R & N)
  - Delete elements designated with "$\neg$"
  - This is *not* logical negation (think about why)

Representations

In Situation Calculus

Δ contains all initial WFFs

No distinction between operators and initial state

Operator definitions distributed throughout Δ

In Strips

Operator information is centralized

Operator information is stored separately

State information is stored separately for each state

No longer need a situation designator

Closed world assumption

Strips Move Operator (?)

Move (x, y, z):

PC: Clr (x), Clr (z), On (x, y), Blk (x),
Diff (x, z), Diff (y, z)

Effects: ¬On (x, y), ¬Clr (z),
On (x, z), Clr (y)

What's wrong?

How can we fix it?

We'd like to say something like:

Move (x, y, z):

PC: Clr (x), Clr (z), On (x, y), Blk (x),
Diff (x, z), Diff (y, z)

Effects: $\neg$On (x, y), Blk (z) $\Rightarrow$ $\neg$Clr (z),
On (x, z), Clr (y)

Now what's wrong?

Need Two Strips Operators

MoveToBlock (x, y, z):

PC: Clr (x), Clr (z), On (x, y), Blk (x),
Blk (z), Diff (x, z), Diff (y, z)

Effects: ¬On (x, y), ¬Clr (z),
On (x, z), Clr (y)

MoveToTable (x, y, z):

PC: Clr (x), On (x, y), Blk (x), Tbl (z), Diff (y, z)

Effects: ¬On (x, y), On (x, z), Clr (y)

Simplifications

We could drop “z”

MoveToTable (x, y, z) to

MoveToTable(x, y) provided…?

Could we leave out “y”?

How about MoveToBlock?

What about Situation Calculus?

Situation Calculus vs STRIPS

Strips Operators do not allow conditional effects

What about Situation Calculus Operators?

Which is more expressive?

Consider a bomb exploding and killing all those around it

There's a set of people near the bomb, each individual in the set is now dead.

PDDL

- Planning Domain Definition Language
- Relax Strips constraints allowing
  - Negations, Conditional effects, Equality
  - Internal quantification, Domain axioms
  - No Closed World Assumption
- Generally requires set of objects to be constant (no cutting blocks in half)
- Often *implemented* as a reduction to Strips operators…
- Example:

```
(define (domain  mcd-blocksworld-axiom)
   (:requirements :adl :domain-axioms :quantified-preconditions)

  (:constants Table)
  (:predicates (on ?x ?y)
             (clear ?x)
             (block ?b)
             (above ?x ?y))

  (:axiom
       :vars (?b ?x)
       :context (or (= ?x Table)
                    (not (exists (?b) (on ?b ?x))))
       :implies (clear ?x))

  (:action puton
         :parameters (?x ?y ?d)
         :precondition (and (not (= ?x ?y)) (not (= ?x table)) (not (= ?d ?y))
                                  (on ?x ?d) (clear ?x) (clear ?y))
         :effect
         (and (on ?x ?y) (not (on ?x ?d))
              (forall (?c)    (when (or (= ?y ?c) (above ?y ?c))
                                  (above ?x ?c)))
              (forall (?e)     (when (and (above ?x ?e) (not (= ?y ?e))
                                             (not (above ?y ?e)))
                                  (not (above ?x ?e)))))))
```

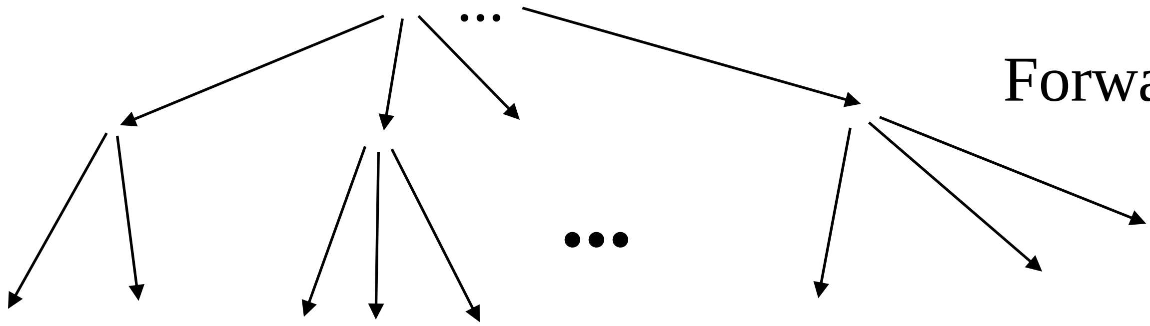
```
(:constants Table)
(:predicates (on ?x ?y)
        (clear ?x)
        (block ?b)
        (above ?x ?y))
```

```
(:axiom
    :vars (?b ?x)
    :context (or (= ?x Table)
                 (not (exists (?b) (on ?b ?x))))
    :implies (clear ?x))
```

```
(:action puton
    :parameters (?x ?y ?d)
    :precondition (and (not (= ?x ?y)) (not (= ?x table)) (not (= ?d ?y))
                       (on ?x ?d) (clear ?x) (clear ?y))
    :effect
    (and (on ?x ?y)        (not (on ?x ?d))
        (forall (?c)        (when (or (= ?y ?c) (above ?y ?c))
                       (above ?x ?c)))
        (forall (?e)        (when (and (above ?x ?e) (not (= ?y ?e))
                                  (not (above ?y ?e)))
                        (not (above ?x ?e)))))))
```

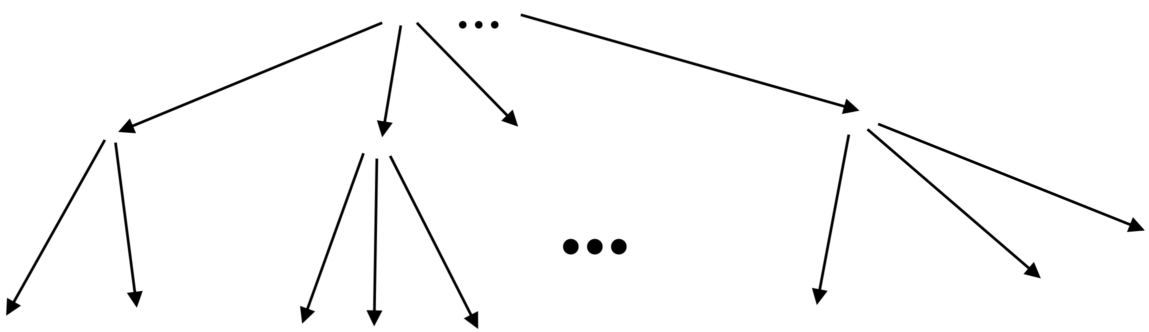
State Space Planner

Initial State



Forward Search for Goal

Goal

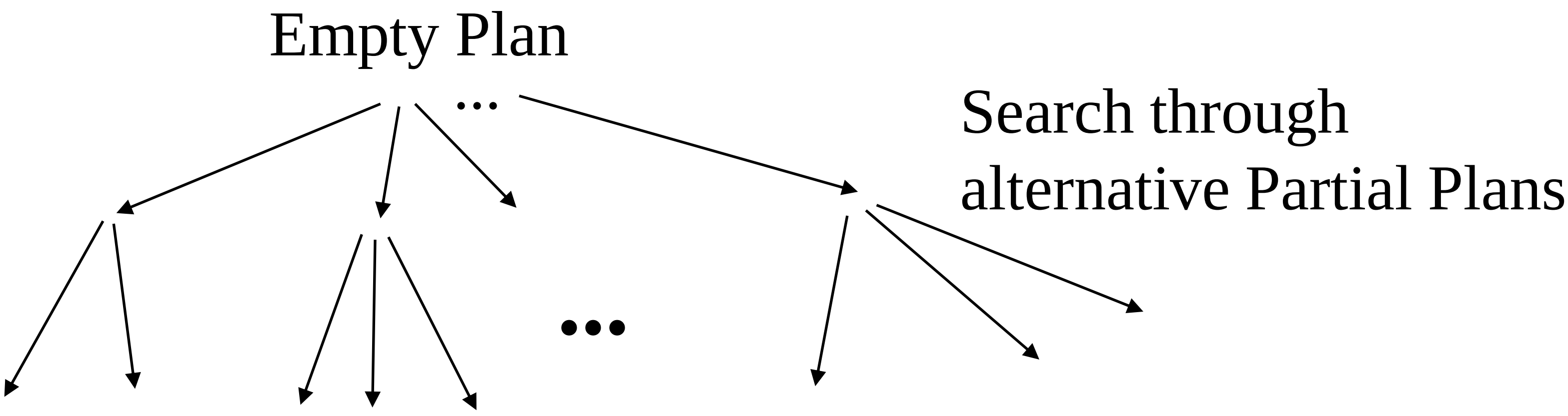


Backward Search for subset of Initial State fluents

One reason why planning beats searching

Plan Space Planner

partial-order planner; nonlinear planner;...



Partial plan ≡ set of constraints

Constraint set denotes all action sequences that satisfy its constraints

Empty plan ≡ all action sequences

Search through alternative constraints for a partial plan that achieves the goal