



# **Hierarchical Task Network (HTN) Planning**

Section 12.2

# Outline

- Example
- Primitive vs. non-primitive operators
- HTN planning algorithm
- Practical planners

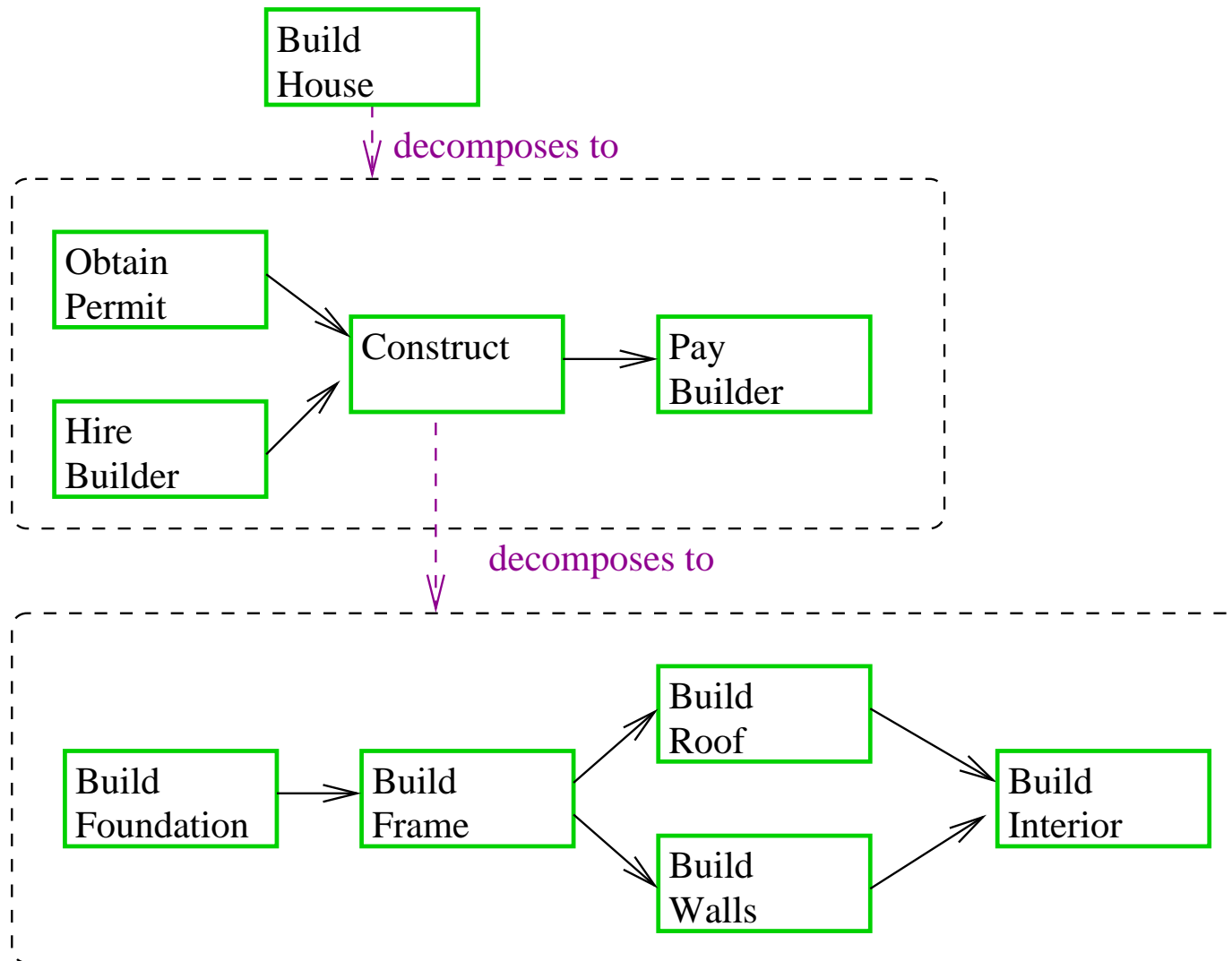
Additional references used for the slides:

**desJardins**, M. (2001). CMSC 671 slides.  
*[www.cs.umbc.edu](http://www.cs.umbc.edu)*

# Hierarchical Task Network (HTN) planning

- Idea: Many tasks in real life already have a built-in hierarchical structure
- For example: a computational task, a military mission, an administrative task
- It would be a waste of time to construct plans from individual operators. Using the built-in hierarchy help escape from exponential explosion
- Running example: the activity of building a house consists of obtaining the necessary permits, finding a builder, constructing the exterior/interior, ...
- HTN approach: use *abstract operators* as well as *primitive operators* during plan generation.

# Building a house



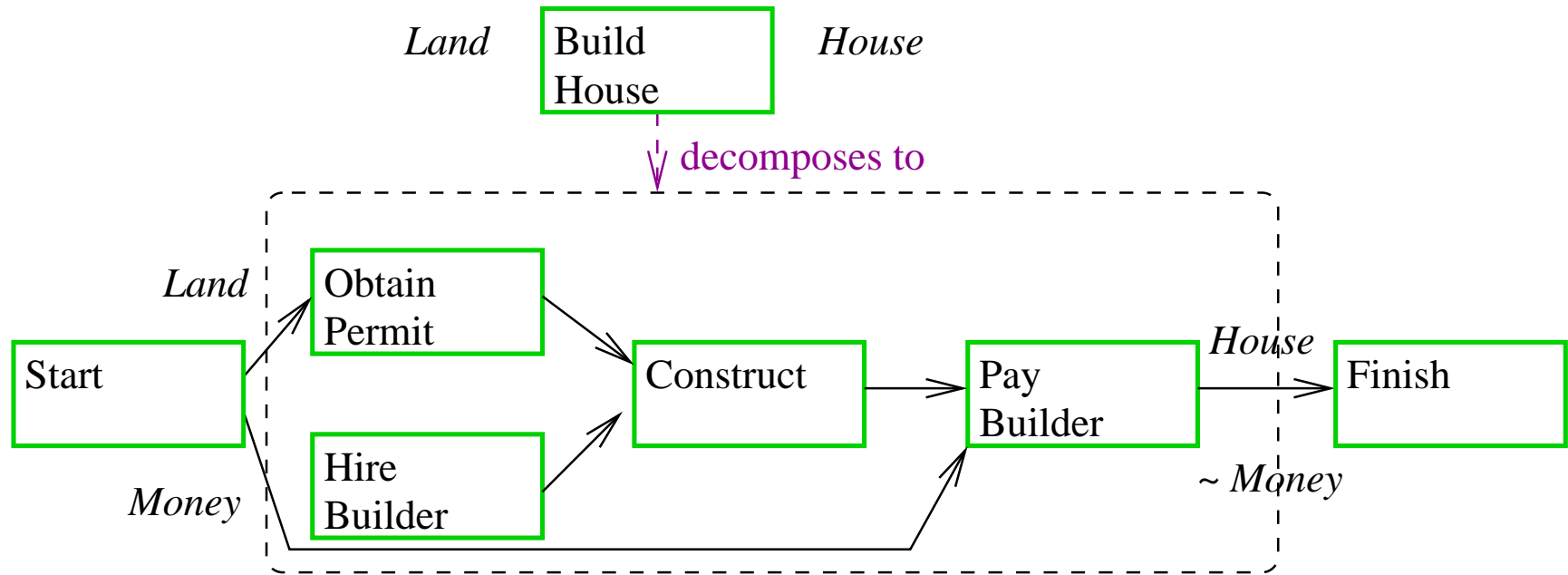
# Hierarchical decomposition

- HTN is suitable for domains where tasks are naturally organized in a hierarchy.
- Uses abstract operators to start a plan.
- Use partial-order planning techniques and *action decomposition* to come up with the final plan
- The final plan contains only *primitive* operators.
- What is to be considered primitive is subjective: what an agent considers as primitive can be another agent's plans.

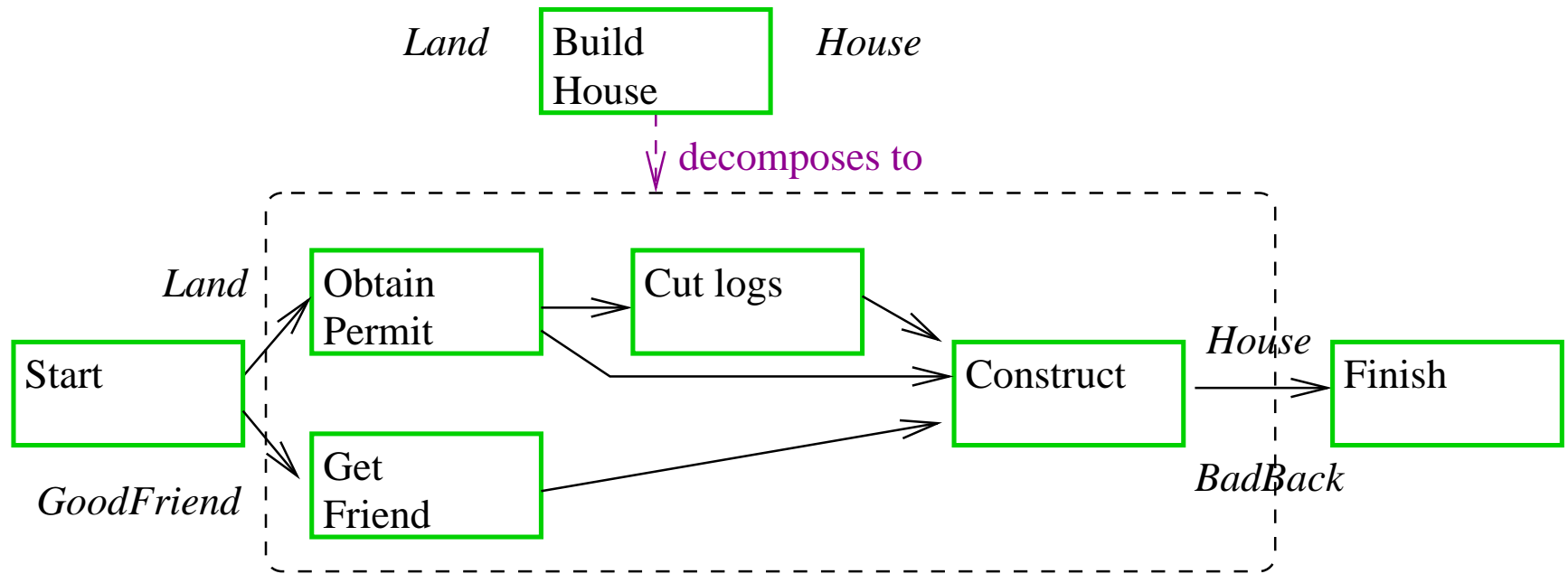
# Representing action decompositions

- A *plan library* contains both primitive and non-primitive actions.
- Non-primitive actions have *external preconditions*, as well as *external effects*.
- Sometimes useful to distinguish between *primary effects* and *secondary effects*.

# Building a house with causal links



# Another way of building a house





# Example action descriptions

*Action(BuyLand, PRECOND:Money, EFFECT: Land  $\wedge$   $\neg$  Money)*

*Action(GetLoan, PRECOND:GoodCredit, EFFECT: Money  $\wedge$  Mortgage)*

*Action(BuildHouse, PRECOND:Land, EFFECT: House)*

*Action(GetPermit, PRECOND:Land, EFFECT: Permit)*

*Action(HireBuilder, EFFECT: Contract)*

*Action(Construct, PRECOND:Permit  $\wedge$  Contract,  
EFFECT: HouseBuilt  $\wedge$   $\neg$  Permit)*

*Action(PayBuilder, PRECOND:Money  $\wedge$  HouseBuilt,  
EFFECT:  $\neg$  Money  $\wedge$  House  $\wedge$   $\neg$  Contract)*

# Example action descriptions

*Decompose(BuildHouse,*  
Plan(STEPS: { $S_1$ : *GetPermit*,  $S_2$ : *HireBuilder*,  
 $S_3$ : *Construction*,  $S_4$ : *PayBuilder*},  
ORDERINGS: { *Start*  $\prec$   $S_1$   $\prec$   $S_2$   $\prec$   $S_3$   $\prec$   $S_4$   $\prec$  *Finish*,  
*Start*  $\prec$   $S_2$   $\prec$   $S_3$  },  
LINKS: { *Start* Land  $S_1$ , *Start* Money  $S_4$ ,  
 $S_1$  Permit  $S_3$ ,  $S_2$  Contract  $S_3$ ,  $S_3$  HouseBuilt  $S_4$ ,  
 $S_4$  House *Finish*,  $S_4$   $\neg$ Money *Finish*}))

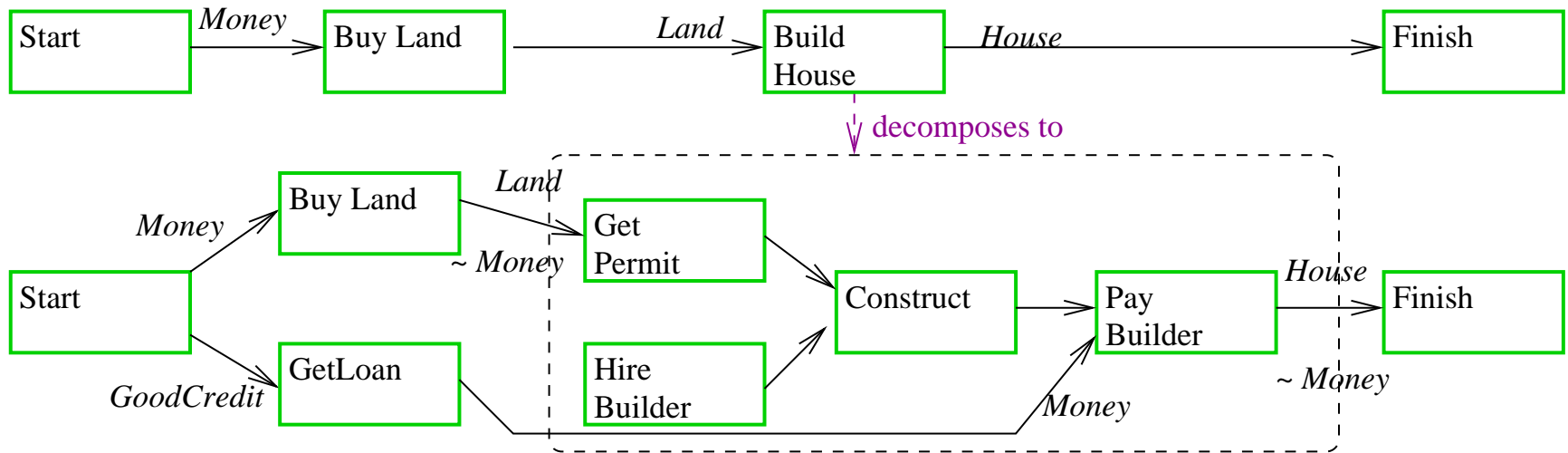
# Correctness

- A decomposition should be a *correct* implementation of the action.
- A plan  $d$  implements an action  $a$  correctly if  $d$  is a complete and consistent partial-order plan for the problem of achieving the effects of  $a$  given the preconditions of  $a$  (result of a **sound** POP).
- The plan library contains several decompositions for any high-level action.
- Each decomposition might have different preconditions and effects. The preconditions of the high-level action should be the **intersection** of the preconditions of the decompositions (similarly for the external effects.)

# Information hiding

- The high-level description hides all the *internal effects* of decompositions (e.g., *Permit* and *Contract*).
- It also hides the duration the internal preconditions and effects hold.
- Advantage: reduces complexity by hiding details
- Disadvantage: conflicts are hidden too

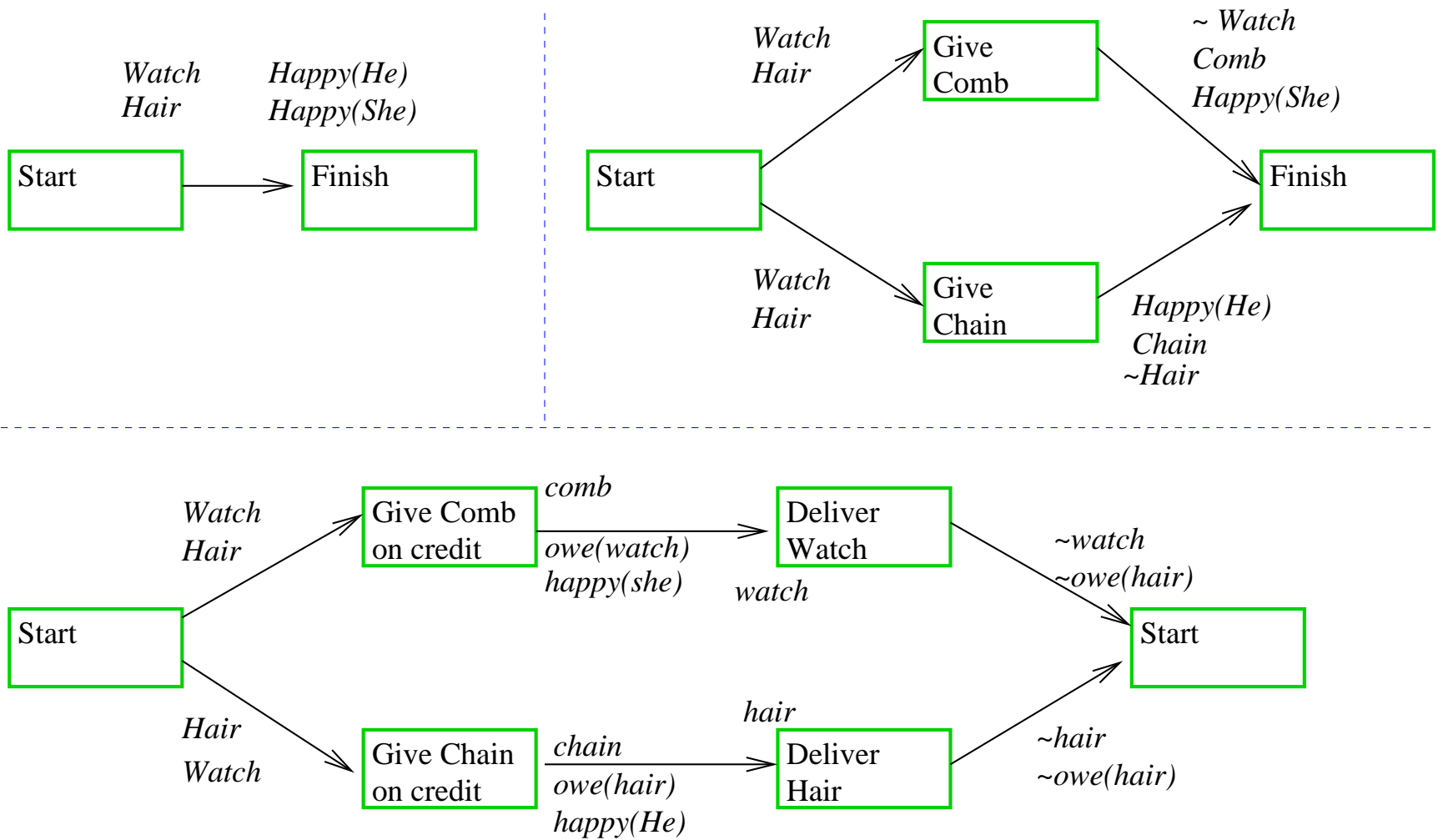
# Example



## For each decomposition $d$ of an action $a$

- Remove the high level action, and insert/reuse actions for each action in  $d$ .  
reuse  $\rightarrow$  *subtask sharing*
- Merge the ordering constraints (If there is an ordering constraint of the form  $B \prec a$ , should every step of  $d$  come after B?)
- Merge the causal links

# Action ordering



# HTN planners

- Most industrial strength planners are HTN based.
- O-PLAN combines HTN planning with scheduling to develop production plans for Hitachi.
- SIPE-2 is an HTN planner with many advanced features
- SHOP is an HTN planner developed at the University of Maryland. It can deal with action durations.



# The features of SIPE-2

- Plan critics
- Resource reasoning
- Constraint reasoning (complex numerical or symbolic variable and state constraints)
- Interleaved planning and execution
- Interactive plan development
- Sophisticated truth criterion
- Conditional effects
- Parallel interactions in partially ordered plans
- Replanning if failures occur during execution

# An operator with constraints

OPERATOR decompose

PURPOSE: Construction

CONSTRAINTS:

Length (Frame)  $\leq$  Length (Foundation),

Strength (Foundation)  $>$  Wt(Frame) + Wt(Roof)  
+ Wt(Walls) + Wt(Interior) + Wt(Contents)

PLOT: Build (Foundation)

Build (Frame)

PARALLEL

Build (Roof)

Build (Walls)

END PARALLEL

Build (Interior)

## More on SIPE-2

- Russell & Norvig explicitly represent causal links; these can also be computed dynamically by using a model of preconditions and effects (this is what SIPE-2 does)
- Dynamically computing causal links means that actions from one operator can safely be interleaved with other operators, and subactions can safely be removed or replaced during plan repair
- Russell & Norvig's representation only includes variable bindings, but more generally we can introduce a wide array of variable constraints

# Truth Criterion

- Determining whether a formula is true at a particular point in a partially ordered plan is, in the general case, NP-hard
- Intuition: there are exponentially many ways to linearize a partially ordered plan
- In the worst case, if there are  $N$  actions unordered with respect to each other, there are  $N!$  linearizations

# Truth Criterion

- Ensuring soundness of the truth criterion requires checking the formula under all possible linearizations
- Use heuristic methods instead to make planning feasible
- Check later to be sure no constraints have been violated

## Truth Criterion in Sipe-2

- Heuristic: prove that there is one possible ordering of the actions that makes the formula true, but don't insert ordering links to enforce that order
- Such a proof is efficient
  - Suppose you have an action A1 with a precondition P
  - Find an action A2 that achieves P (A2 could be initial world state)
  - Make sure there is no action necessarily between A2 and A1 that negates P
- Applying this heuristic for all preconditions in the plan can result in infeasible plans

# Comments on HTN planning

- The major idea is to gain efficiency by using the library of preconstructed plans.
- When there is *recursion*, it is undecidable even if the underlying state space is finite.
  - recursion can be ruled out
  - the length of solutions can be bound
  - can use a hybrid POP and HTN approach

## Comments on HTN planning (cont'd)

- Subtask sharing is nice, but it takes time/resources to notice the opportunities

Would interprocedural optimization be a possibility?  
Consider  $\tan(x) - \sin(x)$ . Both have Taylor series approximations:

$$\tan(x) \approx x + \frac{x^3}{3} + \frac{2x^5}{15} + \frac{17x^7}{315}$$

$$\sin(x) \approx x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$$

It would be nice to share terms but a compiler can only optimize within the code because it does not have the source; and if it did interprocedural optimization  $\tan$  and  $\sin$  would always have to be changed together.



## Comments on HTN planning (cont'd)

- Suppose that we want to construct a plan with  $n$  actions
  - Forward state space planning takes  $O(b^n)$  with  $b$  allowable actions at each state.
  - HTN planning can construct  $d^{(n-1)/(k-1)}$  decomposition trees with  $d$  possible decompositions with  $k$  actions each  
→ keeping  $d$  small and  $k$  large can result in huge savings (long macros usable across a wide range of problems)
  - HTN-based planners do not address uncertainty