Practical Planning

CPSC 470 – Artificial Intelligence

Brian Scassellati

STRIPS planner

STanford Research Institute Problem Solver

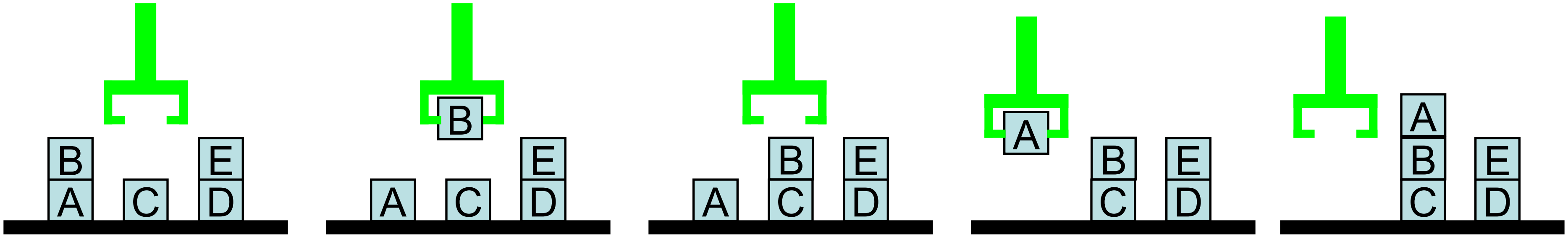
- “Holy Roman Empire” naming
- Represent states and goals in first-order logic

  $At(Home) \wedge Have(Milk) \wedge Have(Drill) \wedge Have(Banana)$

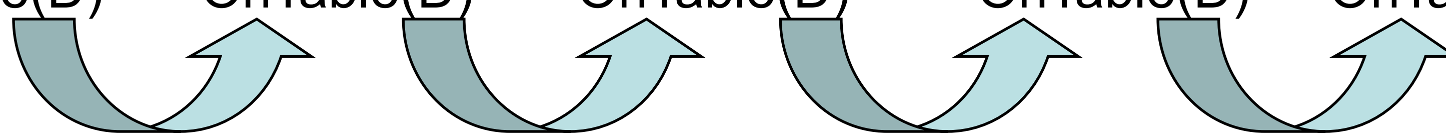
- Assume existential quantification of variables

  $At(x) \wedge Sells(x, Milk)$

Update of Knowledge



Grip(∅)	Grip(B)	Grip(∅)	Grip(A)	Grip(∅)
On(B,A)	TopClear(A)	TopClear(A)	~~TopClear(A)~~	TopClear(A)
TopClear(B)	~~TopClear(B)~~	TopClear(B)	TopClear(B)	~~TopClear(B)~~
OnTable(A)	OnTable(A)	OnTable(A)	~~OnTable(A)~~	On(A,B)
TopClear(C)	TopClear(C)	On(B,C)	On(B,C)	On(B,C)
OnTable(C)	OnTable(C)	OnTable(C)	OnTable(C)	OnTable(C)
On(E,D)	On(E,D)	On(E,D)	On(E,D)	On(E,D)
TopClear(E)	TopClear(E)	TopClear(E)	TopClear(E)	TopClear(E)
OnTable(D)	OnTable(D)	OnTable(D)	OnTable(D)	OnTable(D)

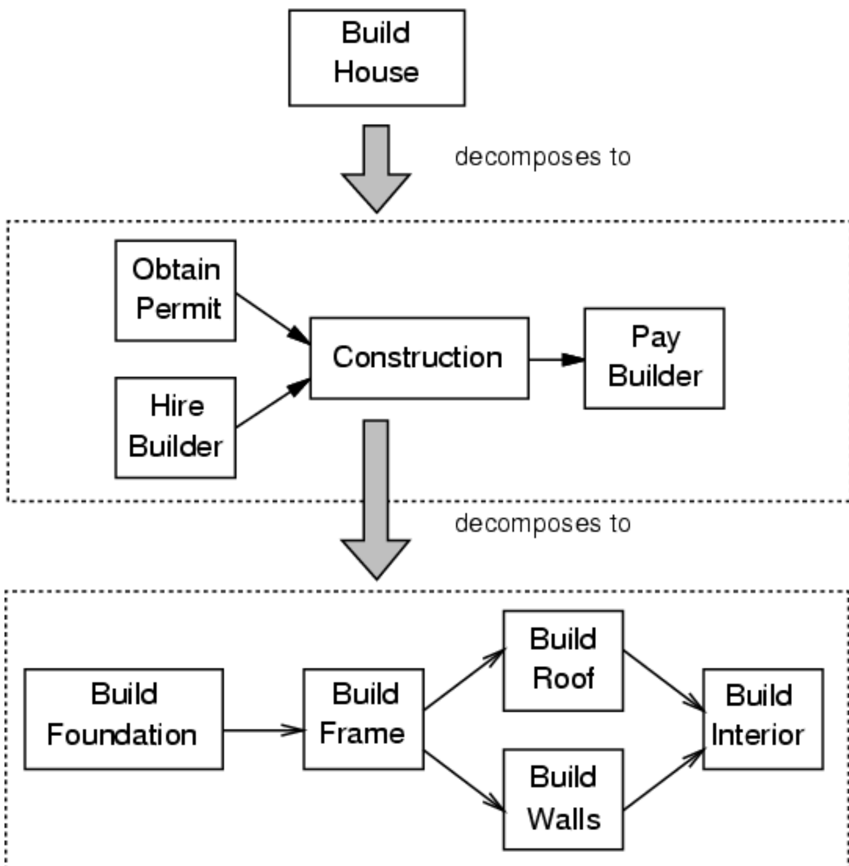


UnStack(B,A) Stack(B,C) Pickup(A) Stack(A,B)

Why STRIPS is Insufficient for many Domains

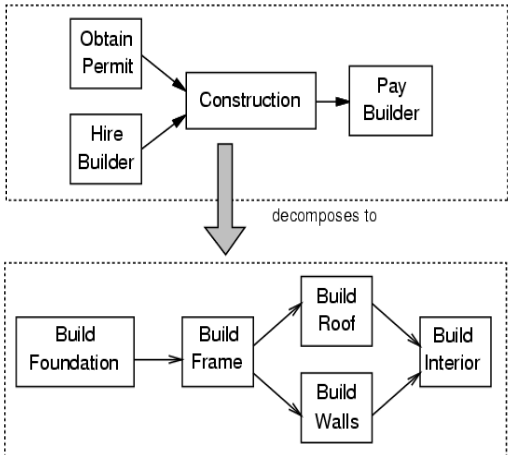
- Hierarchical plans
  - Allow for more complex plans by varying level of abstraction
- Resource Limitations
  - Consumption and generation of resources
  - Time as a resource
    - Based on situation calculus, assumes all actions take place simultaneously and in one unit of time
    - Actions in a plan may have durations, deadlines, and time windows
- Complex conditions
  - No conditionals in STRIPS
  - No universals in STRIPS
- Dealing with incomplete or inaccurate information
  - Conditional planning
  - Execution monitoring

Hierarchical Decomposition



- Primitive and abstract operators
- New decomposition methods
- Describe a decomposition:
  - A set of steps
  - A set of bindings
  - A set of links
  - A set of orderings of steps

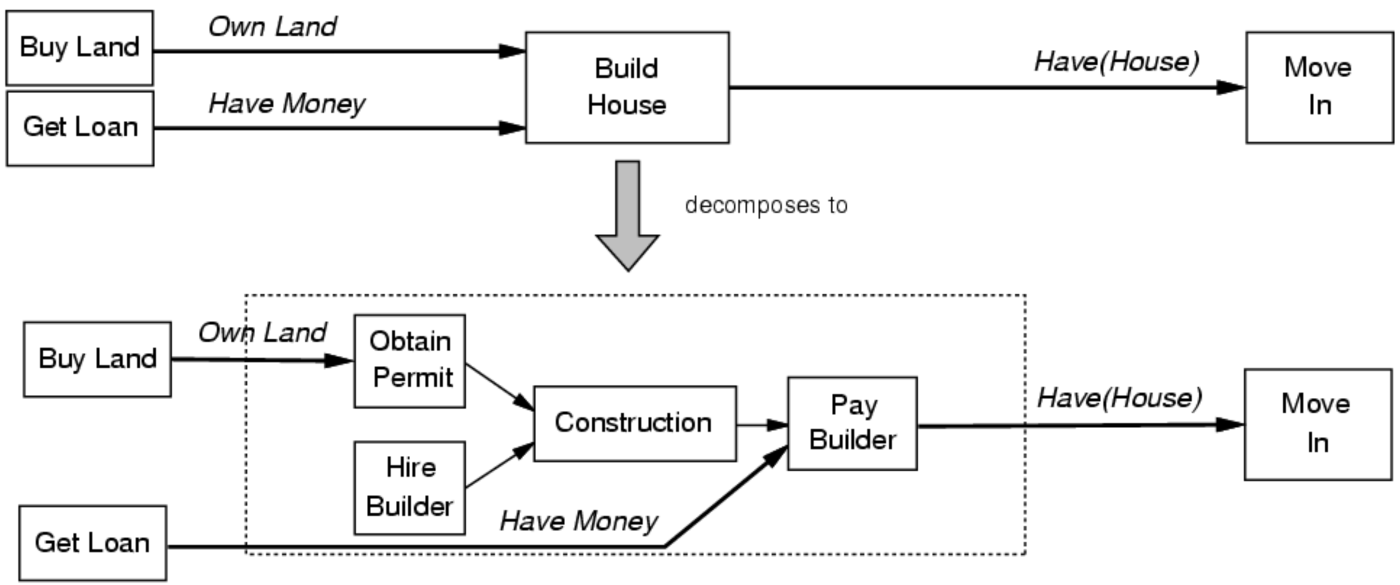
Extending the STRIPS Language to handle Hierarchical Plans



A decomposition is like a subroutine or a macro decomposition for an operator

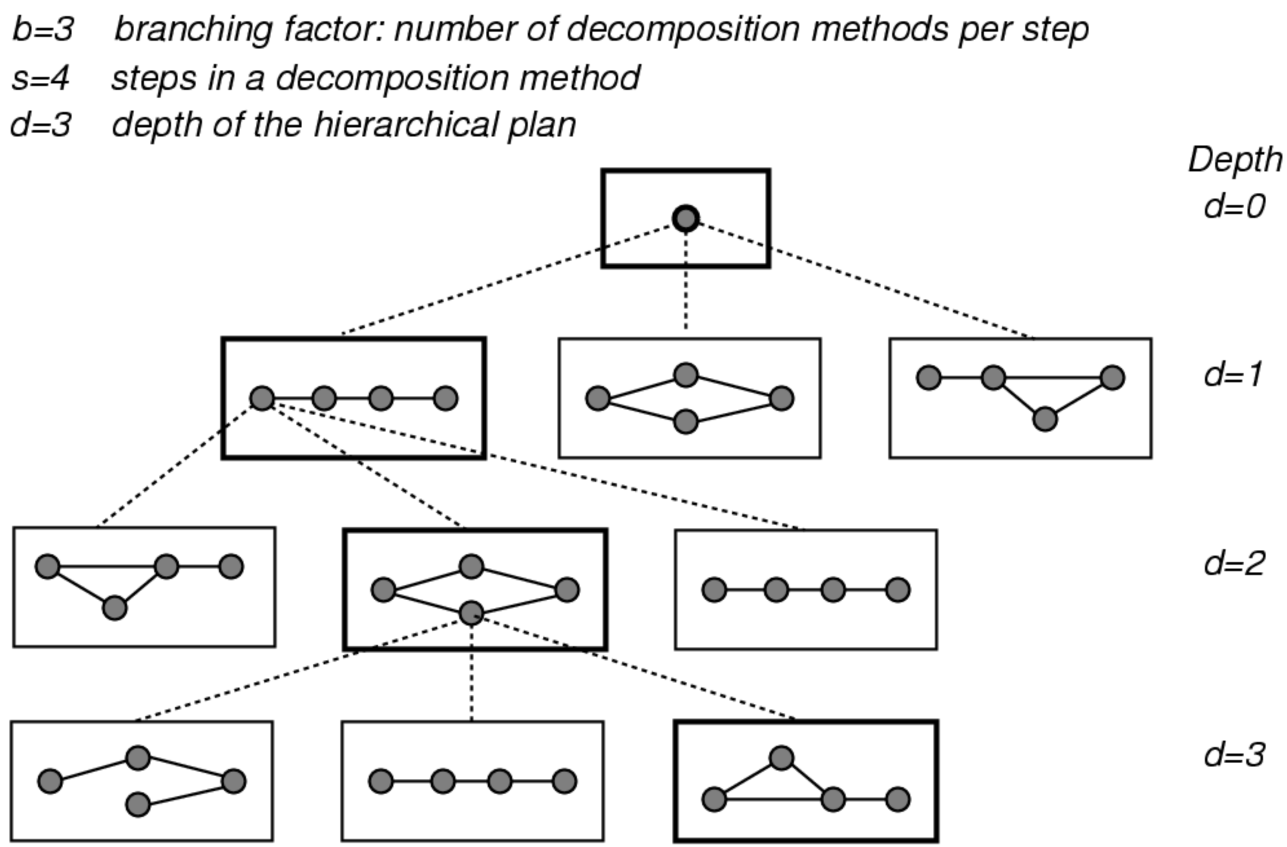
Decompose(Construction,
  Plan(
    Steps: { S1: Build(Foundation), S2: Build(Frame),
        S3: Build(Roof), S4: Build(Walls), S5: Build(Interior)},
    Orderings: {S1 < S2 < S3 < S5, S2 < S4 < S5},
    Bindings: {},
    Links:{S1→S2, S2→S3, S2→S4, S3→S5, S4→S5}
    ))

Hierarchical Decomposition



- Must match the pre-conditions and the post-conditions for each decomposition
- The creation of abstract operators encapsulates the details of creation and leaves only a set of pre- and post- conditions

Search Space for Hierarchical Decomposition



Non-hierarchical planner generates $3x10^{30}$ plans

Hierarchical planner generates 576 plans

Resource Constraints

- Planning with consumables
  - Shopping example
    - Reason about quantities
    - Purchase items
    - Paying in cash
    - Making change
  - STRIPS is not equipped to deal with these types of operations. We must extend the language

Resource Constraints

- Use Measures
  - Quantitative properties of objects like mass, length, and cost

    $Length(Box13)=Meters(1.4)$

    $Price(Orange13)=Cents(20)$
  - Distinguish between amounts and instruments

    $\forall d \;\; d \in Days \Rightarrow Duration(d)=Hours(24)$

    $\forall b \;\; b \in DollarBills \Rightarrow CashValue(b)=\$(1.00)$

Resource Constraints

- Add inequality tests and basic arithmetic operations to the STRIPS language

$At(Store) \wedge InStock(x) \wedge MyCash \geq Price(x,Store)$

Buy(x,Store)

$Have(x) \wedge MyCash \leftarrow MyCash - Price(x,Store)$

$At(GasStation)$

Fillup(GasLevel)

$GasLevel \leftarrow Gallons(15) \wedge MyCash \leftarrow MyCash - (UnitPrice(Gas) \times (Gallons(15) - GasLevel))$

Resource Constraints: Time

- Treat time as just another limited resource
- Some differences
  - Actions executed in parallel consume the maximum of their respected times
    - (as opposed to money, in which parallel actions consume the sum)
  - Time constraints must be consistent with ordering constraints
  - Time can never move backward

Complex Conditions

- STRIPS representation
  - Represent states and goals in first-order logic
    At(Home) ∧ Have(Milk) ∧ Have(Drill) ∧ Have(Banana)
  - Assume existential quantification of variables
    At(x) ∧ Sells(x, Milk)
- We will sometime require more complex operators for real-world applications
  - Conditional effects
  - Universal quantification
  - Negated goals
  - Disjunctive goals
- These additions will require both changes to the representation language and to the theorem prover

Complex Conditions

Conditional Effects

Move(b,x,y)

Precond: On(b,x) $\wedge$ Clear(b) $\wedge$ Clear(y)

Effect: On(b,y) $\wedge$ Clear(x) $\wedge$ $\neg$On(b,x) $\wedge$ $\neg$Clear(y) when y $\neq$ Table

"Move block b from x to y"

Include in the effect that y now becomes clear except when y is the table

Universal Quantification

Carry(bag,x,y)

Precond: Bag(bag) $\wedge$ At(bag,x)

Effect: At(bag,y), $\neg$ At(bag,x) $\wedge$ $\forall$i Item(i) $\Rightarrow$ (At(i,y) $\wedge$ $\neg$At(i,x)) when In(i,bag)

"When you carry a bag from x to y, all items in the bag at x are now at y"

Allows us to define the rules of movement without listing each individual object

Complex Conditions

- Negated and Disjunctive Goals
  - Disjunctive preconditions
    - Can perform an action if either ***p*** or ***q***
    - Relatively easy to change syntax
    - Relatively easy to change theorem prover
  - Disjunctive Effects
    - Action results in either effect ***p*** or ***q***
    - Relatively easy to change syntax
    - Relatively hard to change theorem prover

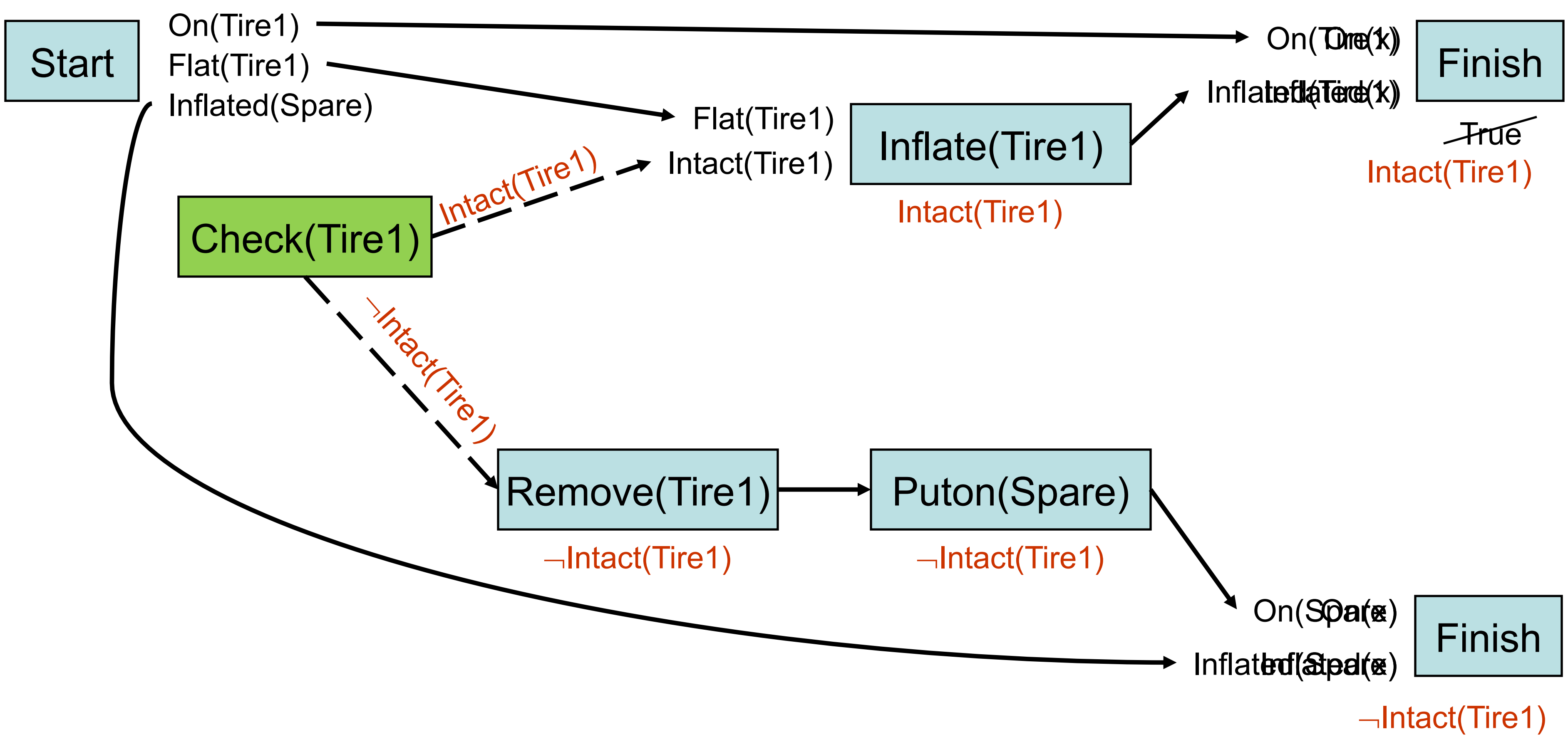
Incomplete or Inaccurate Information

- Problems can evolve from
  - Sensory failures
  - Execution errors
  - Flawed planning
  - Inaccessible world information
- Two methods for addressing this
  - Conditional planning
  - Execution monitoring

Conditional Planning

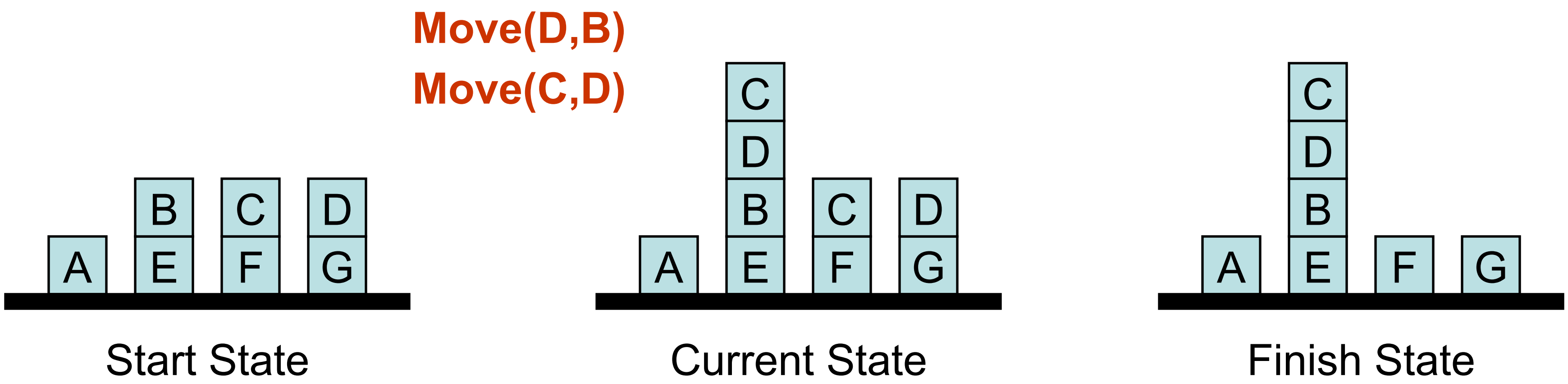
- Deals with incomplete information by constructing a plan that accounts for alternate situations/contingencies
- Agent executes sensing actions to test appropriate conditions
- Simple example
  - Shopping agent
    - Check price to see if it exceeds current cash

Conditional Planning Example

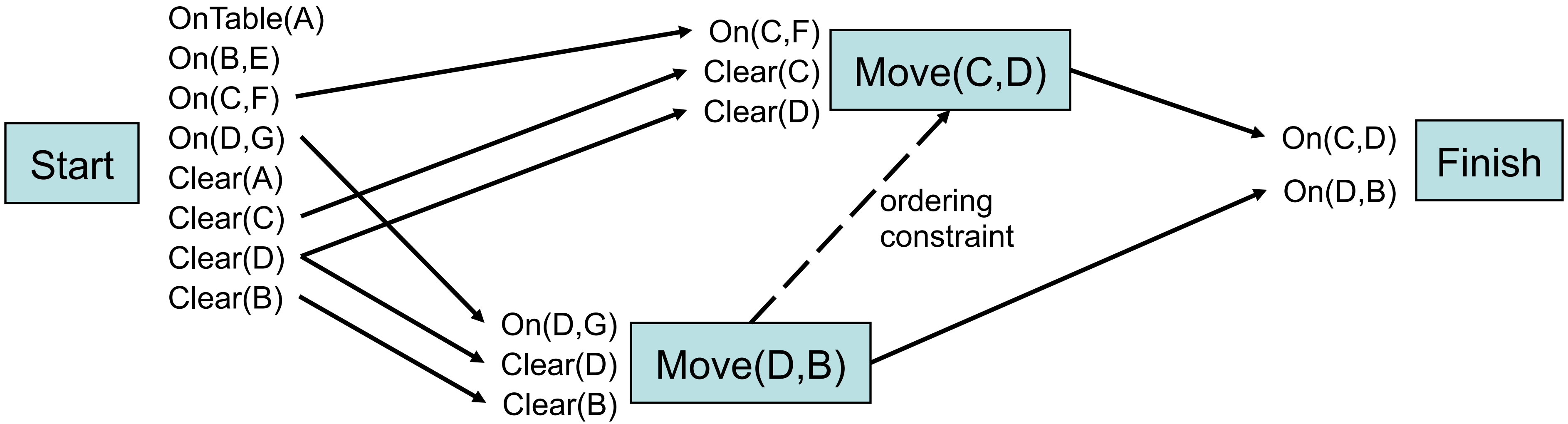


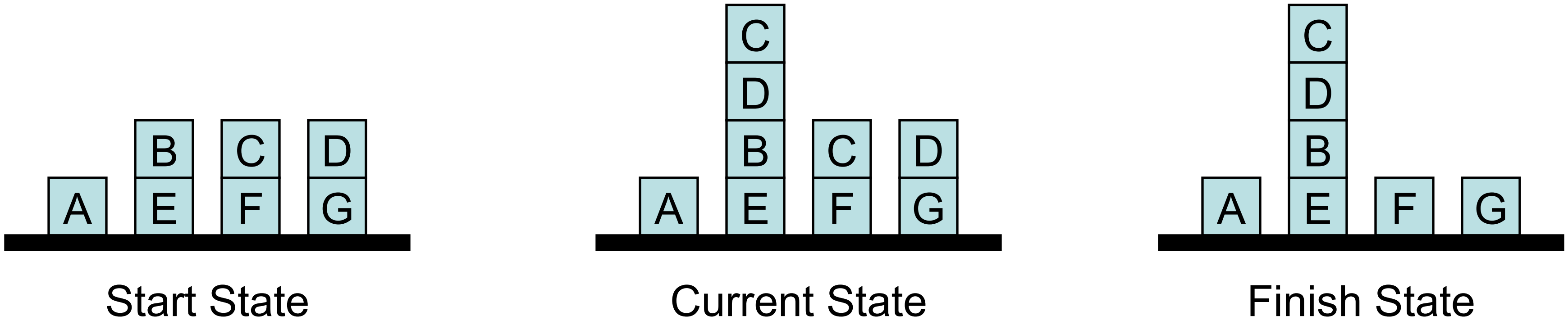
Execution Monitoring

- Monitor what is happening while a plan is executing
  - Provides meaningful description of state throughout execution
  - Monitors for errors in perception and execution
- Blocks world example

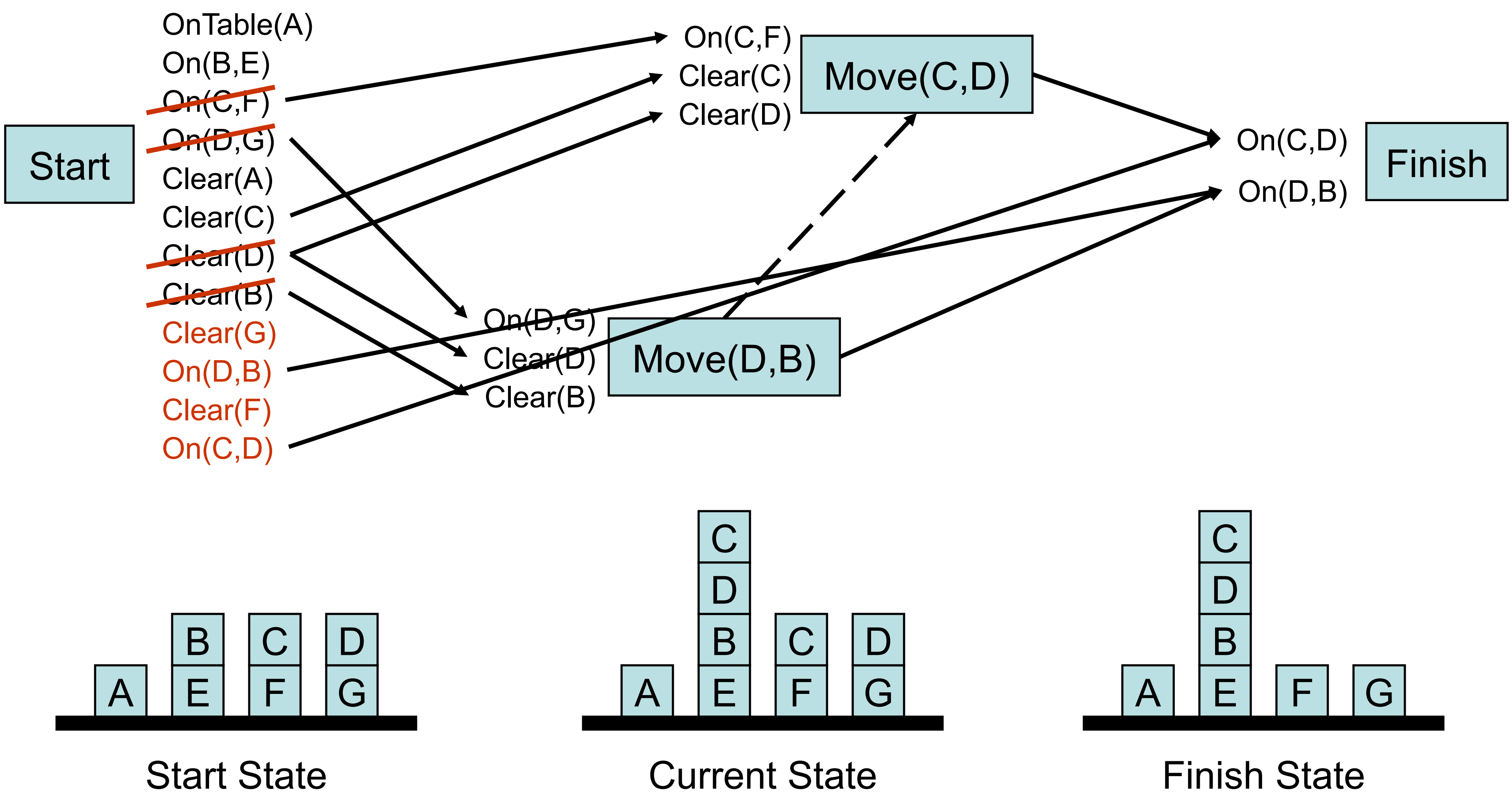


Building a Blocks World Plan

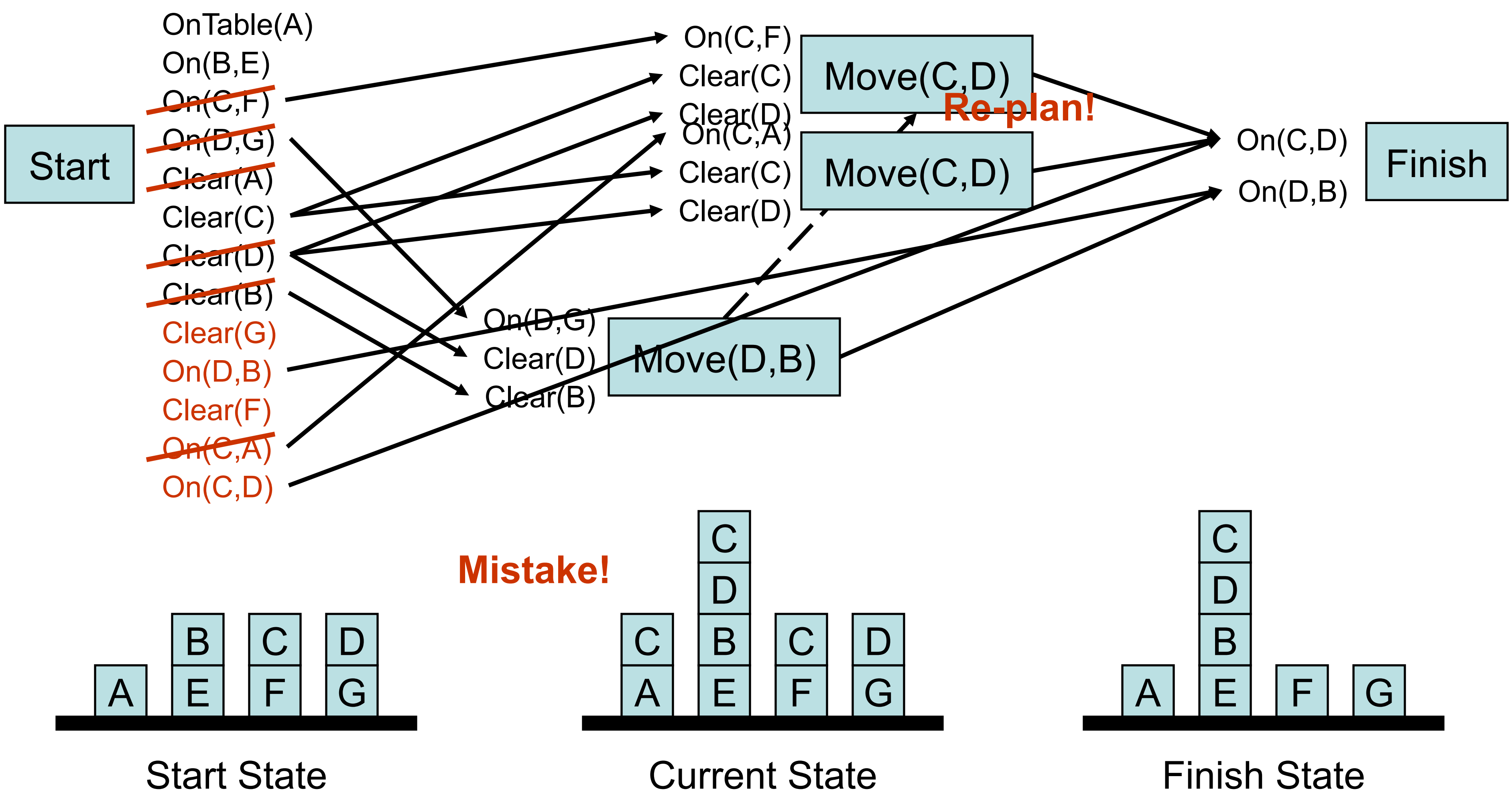




Executing a Blocks World Plan (while maintaining a future plan)

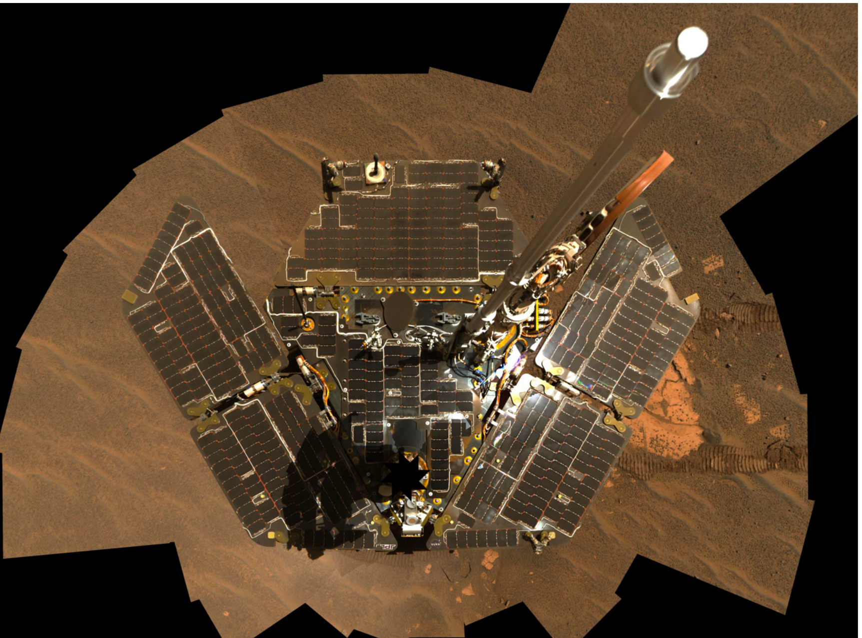


Executing a Blocks World Plan (mistakes happen…)



Planning in Real-World Systems:

Mars Rover Opportunity



- Launched: July 7, 2003
- Time delay 4-30 minutes
- Designed for
  - 90 Martian days
  - 1000m travel
- Active deployment
  - 5,111 Martian days
  - 45,000m travel
  - returned 217,000 images
  - discovered hematite, a mineral formed in water

Administrivia

- Friday: Reasoning with Uncertainty
- PS 4 out today, due next Wednesday.