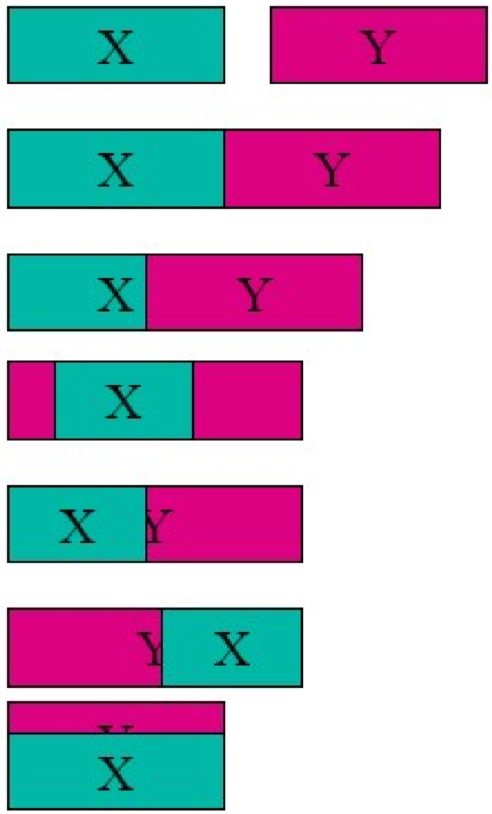
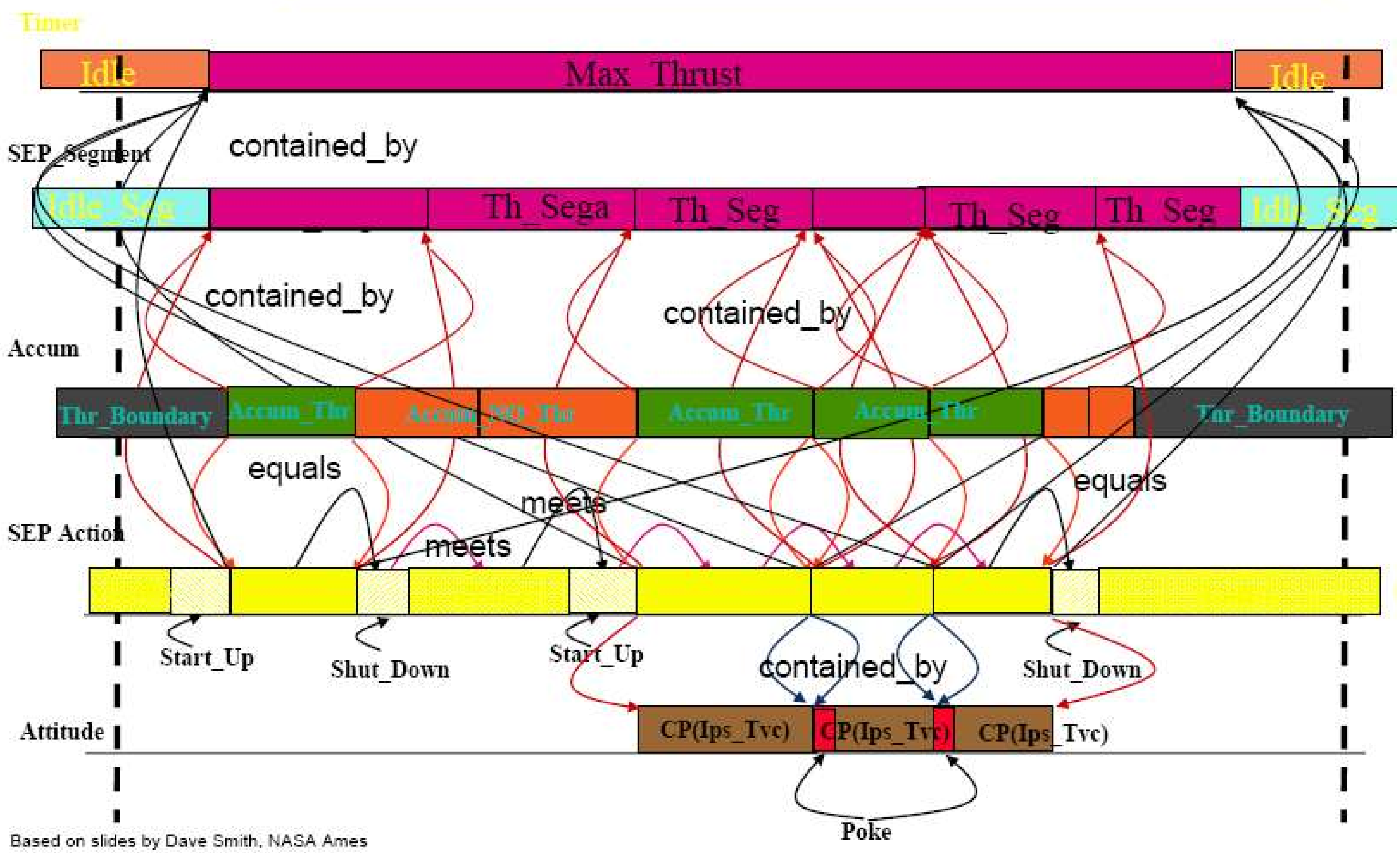
Temporal Constraints

- x before y
- x meets y
- x overlaps y
- x during y
- x starts y
- x finishes y
- x equals y



- y after x
- y met-by x
- y overlapped-by x
- y contains x
- y started-by x
- y finished-by x
- y equals x

RAX Example: DS1



Based on slides by Dave Smith, NASA Ames

Temporal Constraints as Inequalities

- x before y $X^+ < Y^-$
- x meets y $X^+ = Y^-$
- x overlaps y $(Y^- < X^+)$ & $(X^- < Y^+)$
- x during y $(Y^- < X^-)$ & $(X^+ < Y^+)$
- x starts y $(X^- = Y^-)$ & $(X^+ < Y^+)$
- x finishes y $(X^- < Y^-)$ & $(X^+ = Y^+)$
- x equals y $(X^- = Y^-)$ & $(X^+ = Y^+)$

Inequalities may be expressed as binary interval relations:

$$X^+ - Y^- < [-inf, 0]$$

Metric Constraints

- Going to the store takes at least 10 minutes and at most 30 minutes.
  → $10 \leq [T^{+}(\text{store}) - T^{-}(\text{store})] \leq 30$
- Bread should be eaten within a day of baking.
  → $0 \leq [T^{+}(\text{baking}) - T^{-}(\text{eating})] \leq 1$ day
- Inequalities, $X^{+} < Y^{-}$, may be expressed as binary interval relations:
  → $-\inf < [X^{+} - Y^{-}] < 0$

Temporal Constraint Networks

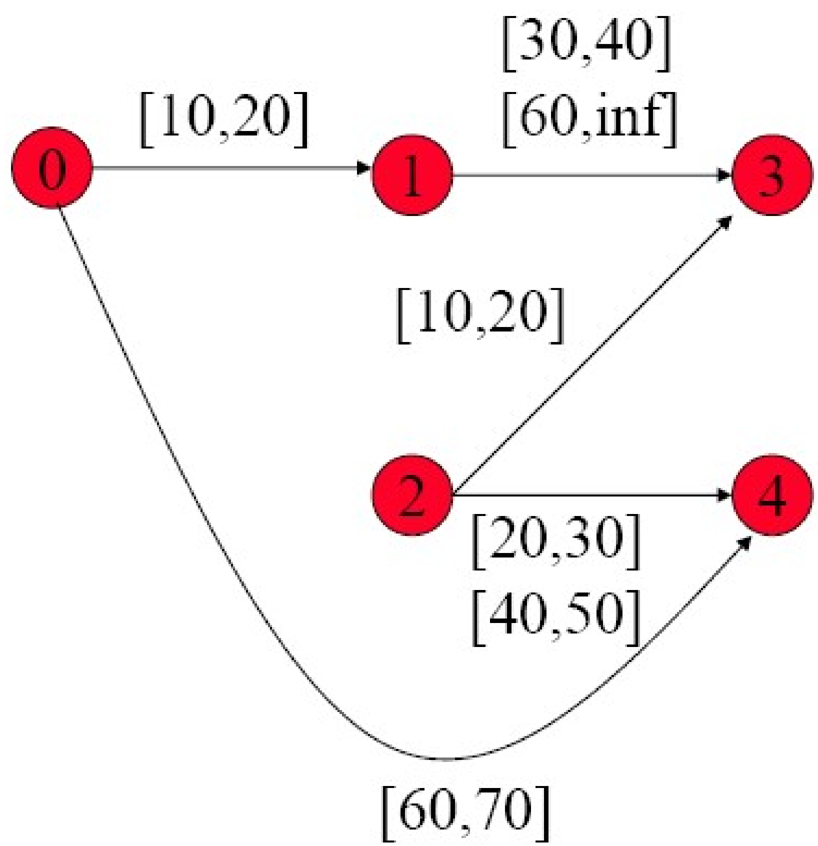
- A set of time points X_i at which events occur.
- Unary constraints

$$(a_0 \leq X_i \leq b_0) \text{ or } (a_1 \leq X_i \leq b_1) \text{ or } \ldots$$

- Binary constraints

$$(a_0 \leq X_j - X_i \leq b_0) \text{ or } (a_1 \leq X_j - X_i \leq b_1) \text{ or } \ldots$$

Temporal Constraint Satisfaction Problem



Simple Temporal Networks

Simple Temporal Networks:

- A set of time points X_i at which events occur.
- Unary constraints

  $(a_0 \le X_i \le b_0)$ ~~or $(a_1 \le X_i \le b_1)$ or . . .~~
- Binary constraints

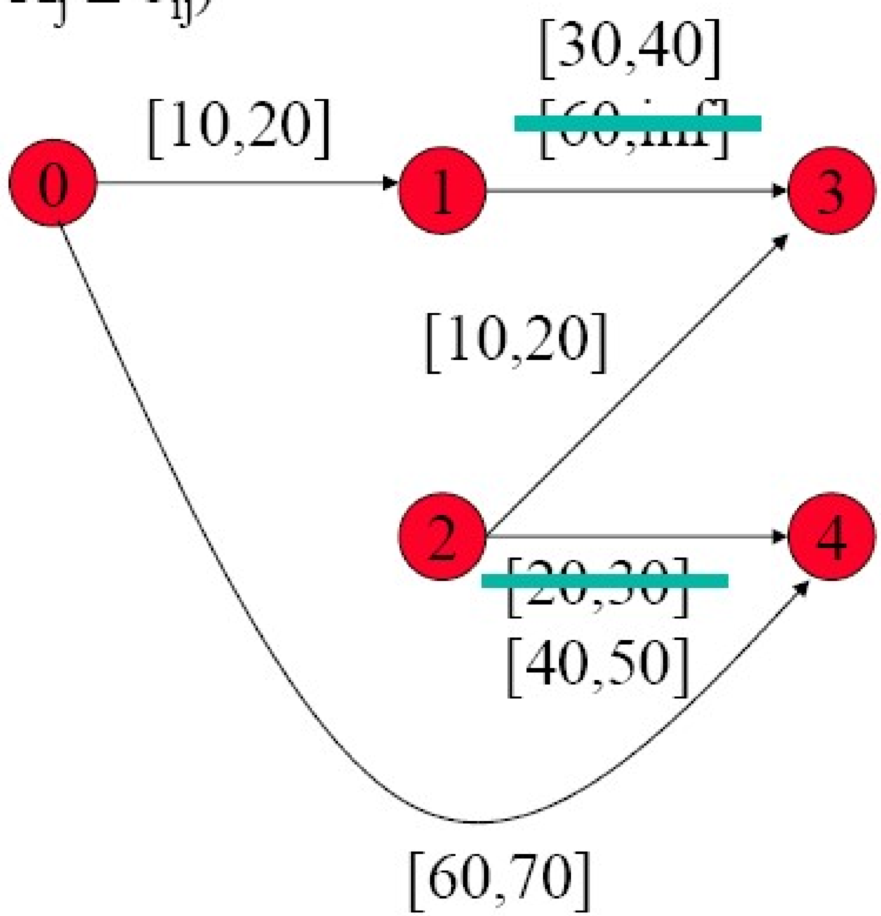
  $(a_0 \le X_j - X_i \le b_0)$ ~~or $(a_1 \le X_j - X_i \le b_1)$ or . . .~~

Sufficient to represent:
- most Allen relations
- simple metric constraints

Can't represent:
- Disjoint activities

Simple Temporal Networks

- $T_{ij} = (a_{ij} \leq X_i - X_j \leq b_{ij})$



Based on slides by Dave Smith, NASA Ames

TCSP Queries

(Dechter, Meiri, Pearl, AIJ91)

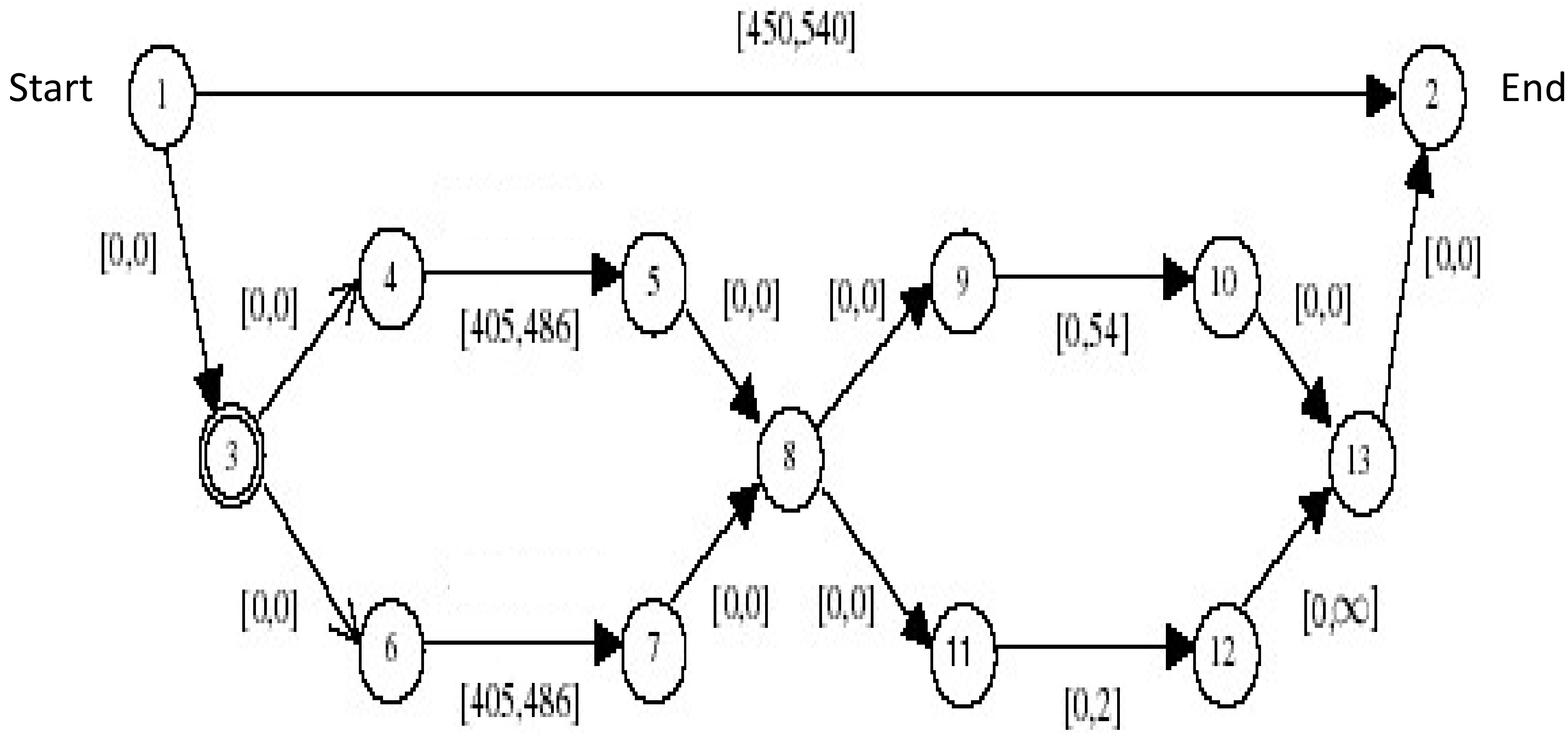
- Is the TCSP consistent?
- What are the feasible times for each X_i?
- What are the feasible durations between each X_i and X_j?
- What is a consistent set of times?
- What are the earliest possible times?
- What are the latest possible times?

TCSP Queries

(Dechter, Meiri, Pearl, AIJ91)

- Is the TCSP consistent? *Planning*
- What are the feasible times for each X_i?
- What are the feasible durations between each X_i and X_j?
- What is a consistent set of times?
- What are the earliest possible times? *Execution*
- What are the latest possible times?

STN example



Start
End
[450,540]
1
2
3
4
5
6
7
8
9
10
11
12
13
[0,0]
[0,0]
[405,486]
[0,0]
[0,0]
[0,54]
[0,0]
[0,0]
[0,0]
[405,486]
[0,0]
[0,0]
[0,2]
[0,∞]

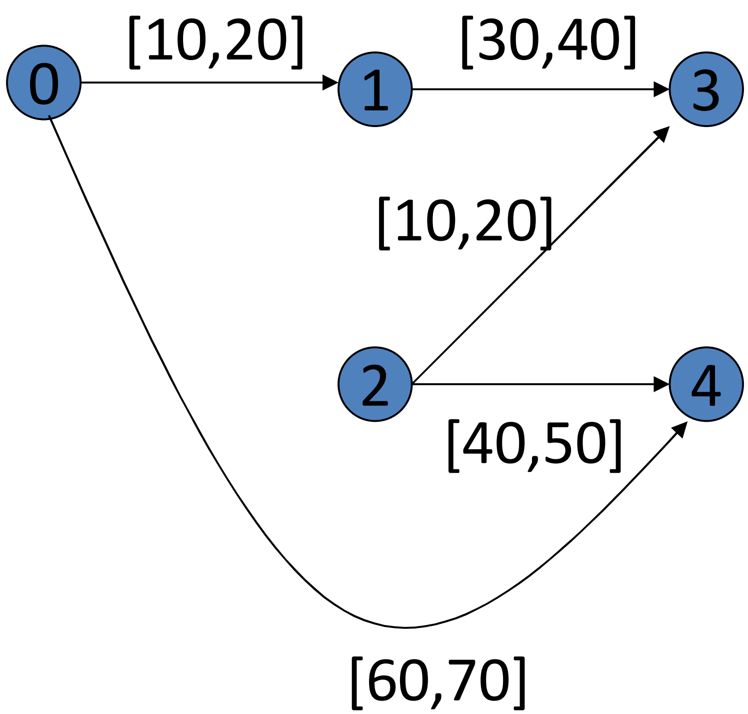
To Query STN Map to Distance Graph $G_d = < V,E_d >$

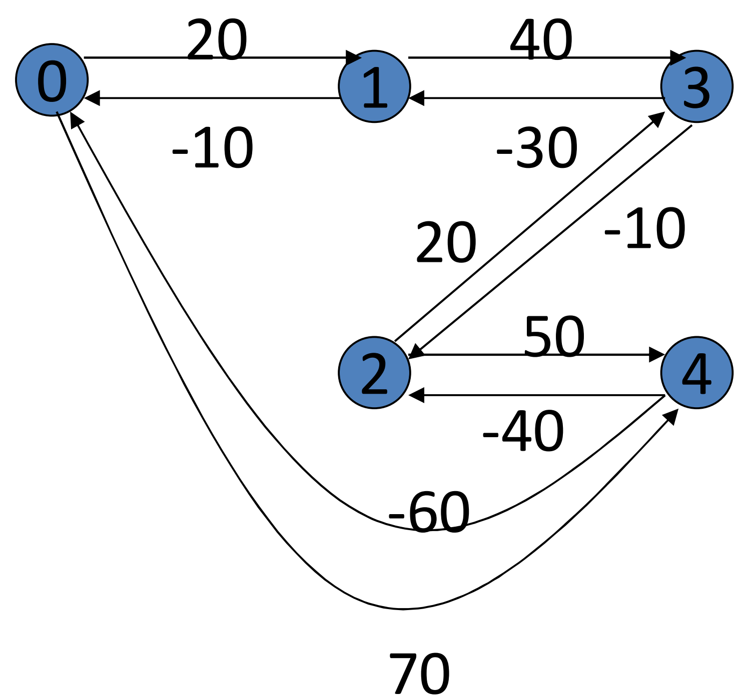
Edge encodes an upper bound on distance to target from source.

$$T_{ij} = (a_{ij} \leq X_j - X_i \leq b_{ij})$$

$$X_j - X_i \leq b_{ij}$$

$$X_i - X_j \leq - a_{ij}$$





Induced Constraints for G_d

constraint: $i_0 = i, i_1 = \ldots, i_k = j$

$$X_j - X_i \leq \sum_{j=1}^{k} a_{i_{j-1}, i_j}$$

→Intersected path constraints:

$$X_j - X_i \leq d_{ij}$$

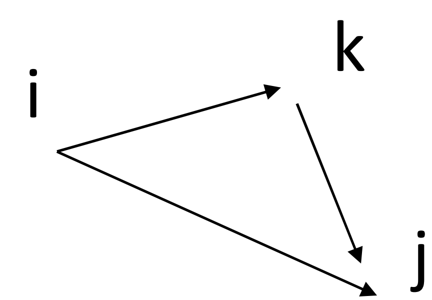
where d_{ij} is the shortest path from i to j

Compute Intersected Paths by All Pairs Shortest Path

(e.g., Floyd-Warshall's algorithm)

1. for i := 1 to n do $d_{ii} \leftarrow 0$;
2. for i, j := 1 to n do $d_{ij} \leftarrow a_{ij}$;

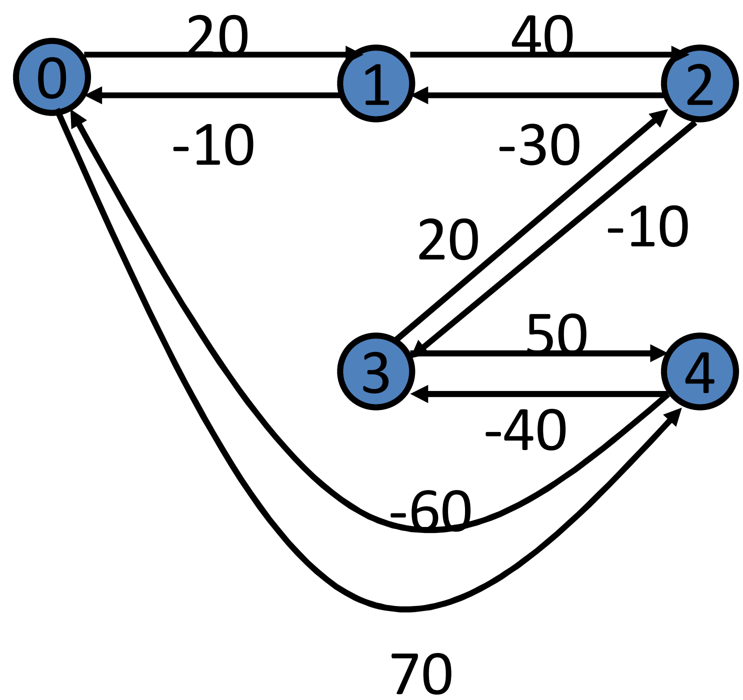
3. for k := 1 to n do
4. for i, j := 1 to n do
5. $d_{ij} \leftarrow \min\{d_{ij}, d_{ik} + d_{kj}\}$;



Shortest Paths of G_d

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph



STN Minimum Network

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

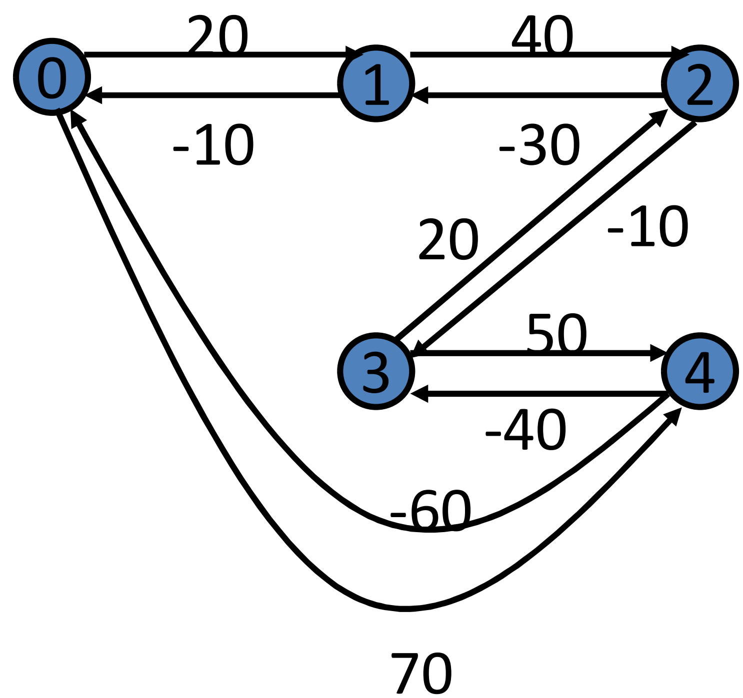
	0	1	2	3	4
0	[0]	[10,20]	[40,50]	[20,30]	[60,70]
1	[-20,-10]	[0]	[30,40]	[10,20]	[50,60]
2	[-50,-40]	[-40,-30]	[0]	[-20,-10]	[20,30]
3	[-30,-20]	[-20,-10]	[10,20]	[0]	[40,50]
4	[-70,-60]	[-60,-50]	[-30,-20]	[-50,-40]	[0]

STN minimum network

Test Consistency:
No Negative Cycles

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

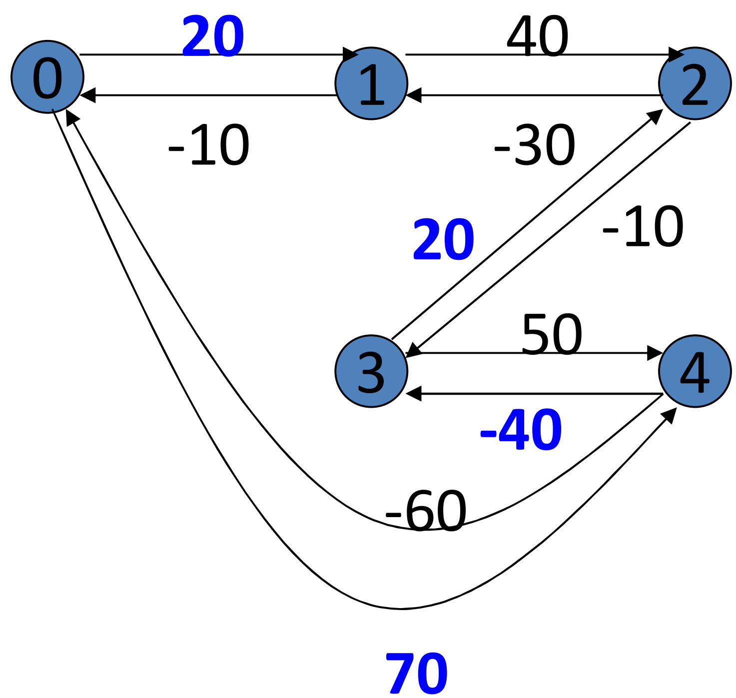


Latest Solution

Node 0 is the reference.

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

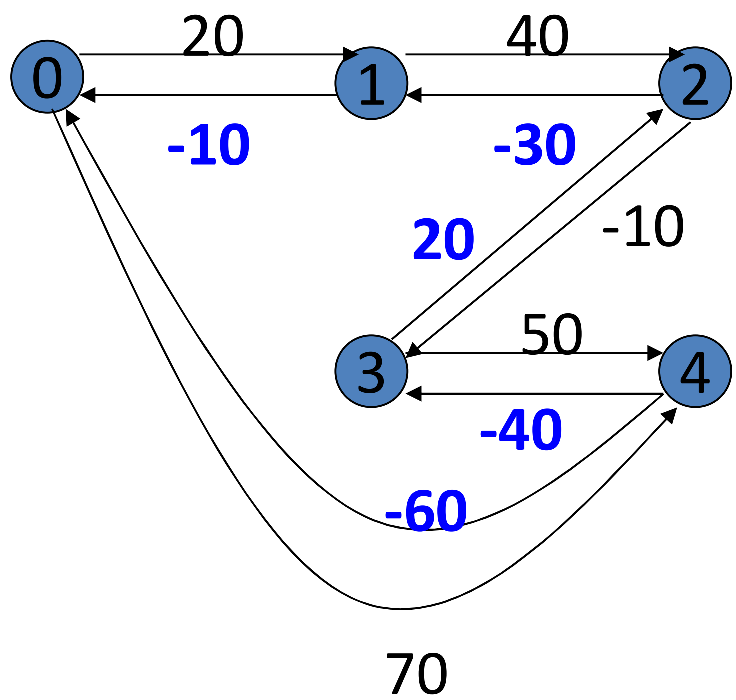


Earliest Solution

Node 0 is the reference.

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph



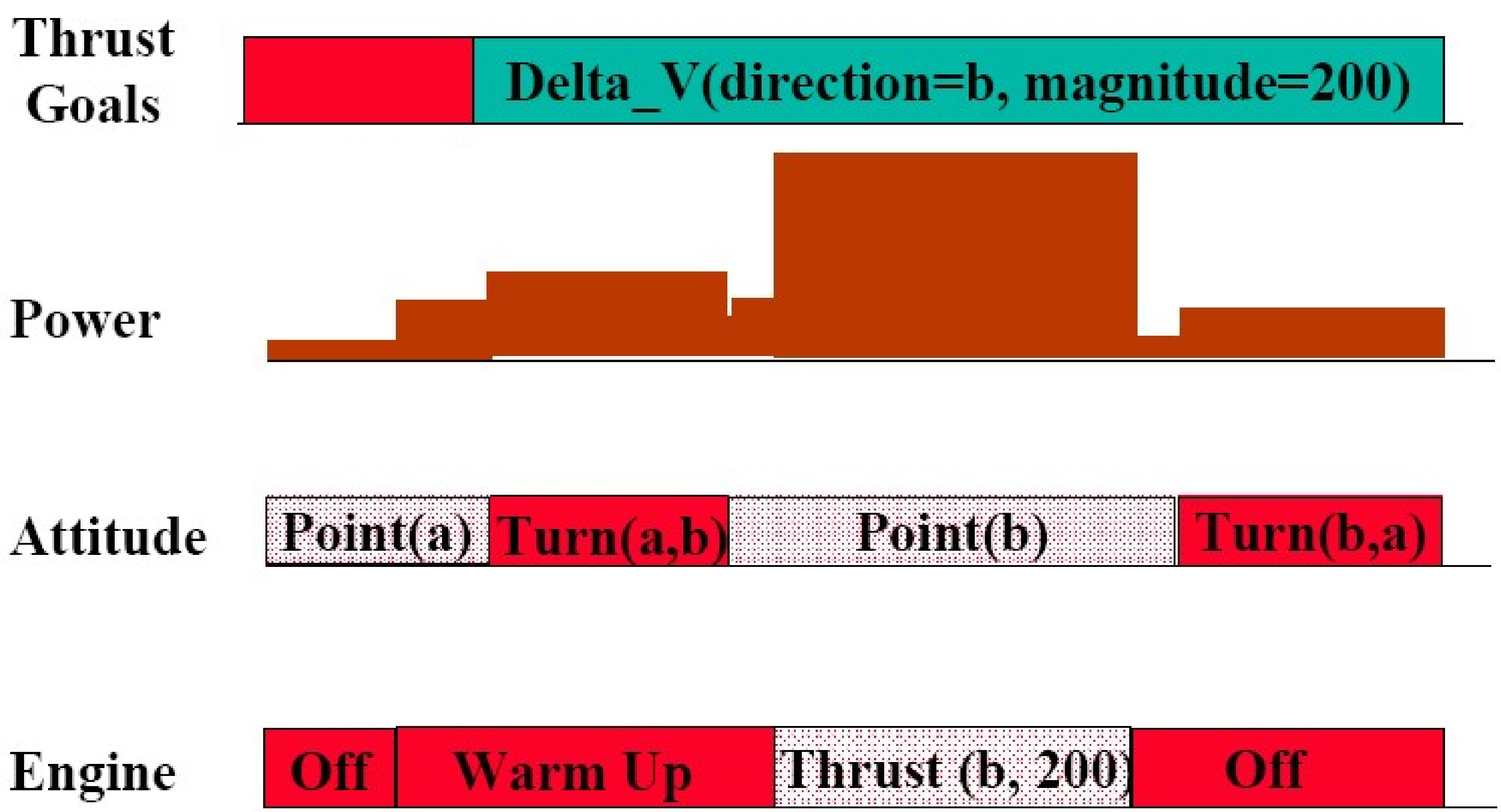
Feasible Values

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

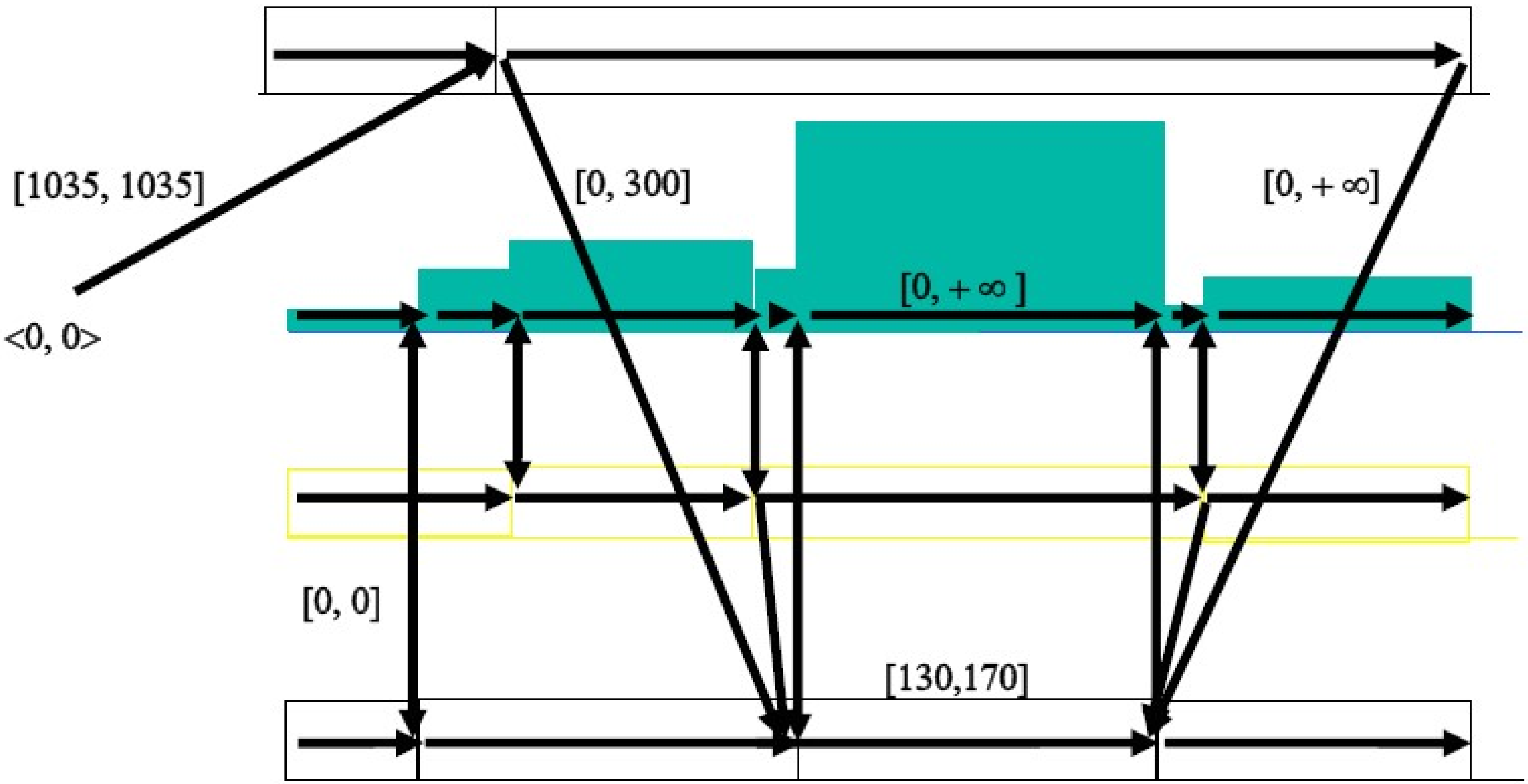
- X_1 in [10, 20]
- X_2 in [40, 50]
- X_3 in [20, 30]
- X_4 in [60, 70]

A Complete CBI-Plan is a STN



Based on slides by Dave Smith, NASA Ames

A Complete CBI-Plan is a STN



Based on slides by Dave Smith, NASA Ames

DS1: Remote Agent

Remote Agent Experiment: RAX

Remote Agent Experiment

See rax.arc.nasa.gov

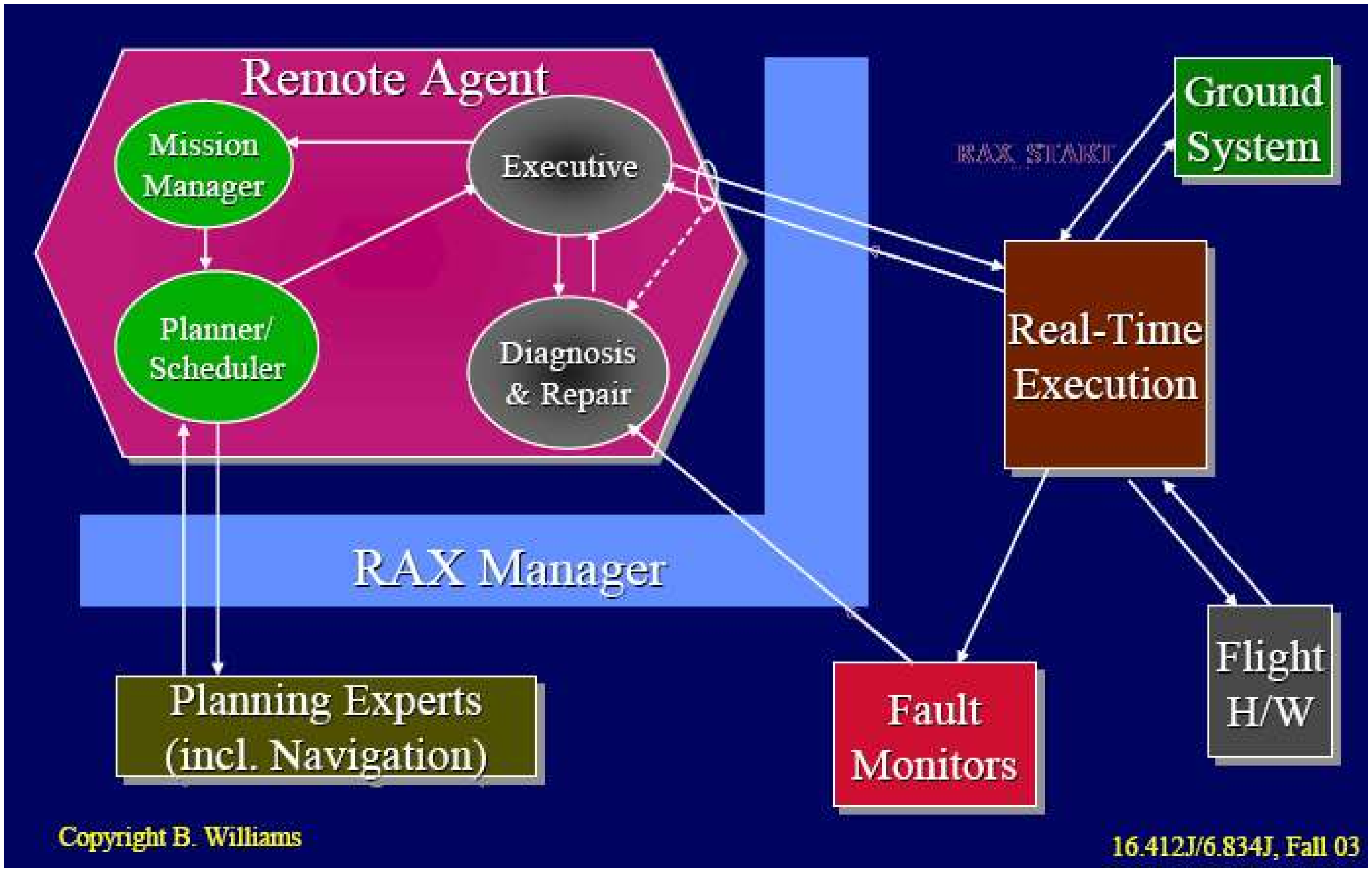
May 17-18th experiment

- Generate plan for course correction and thrust
- Diagnose camera as stuck on
  - Power constraints violated, abort current plan and replan
- Perform optical navigation
- Perform ion propulsion thrust

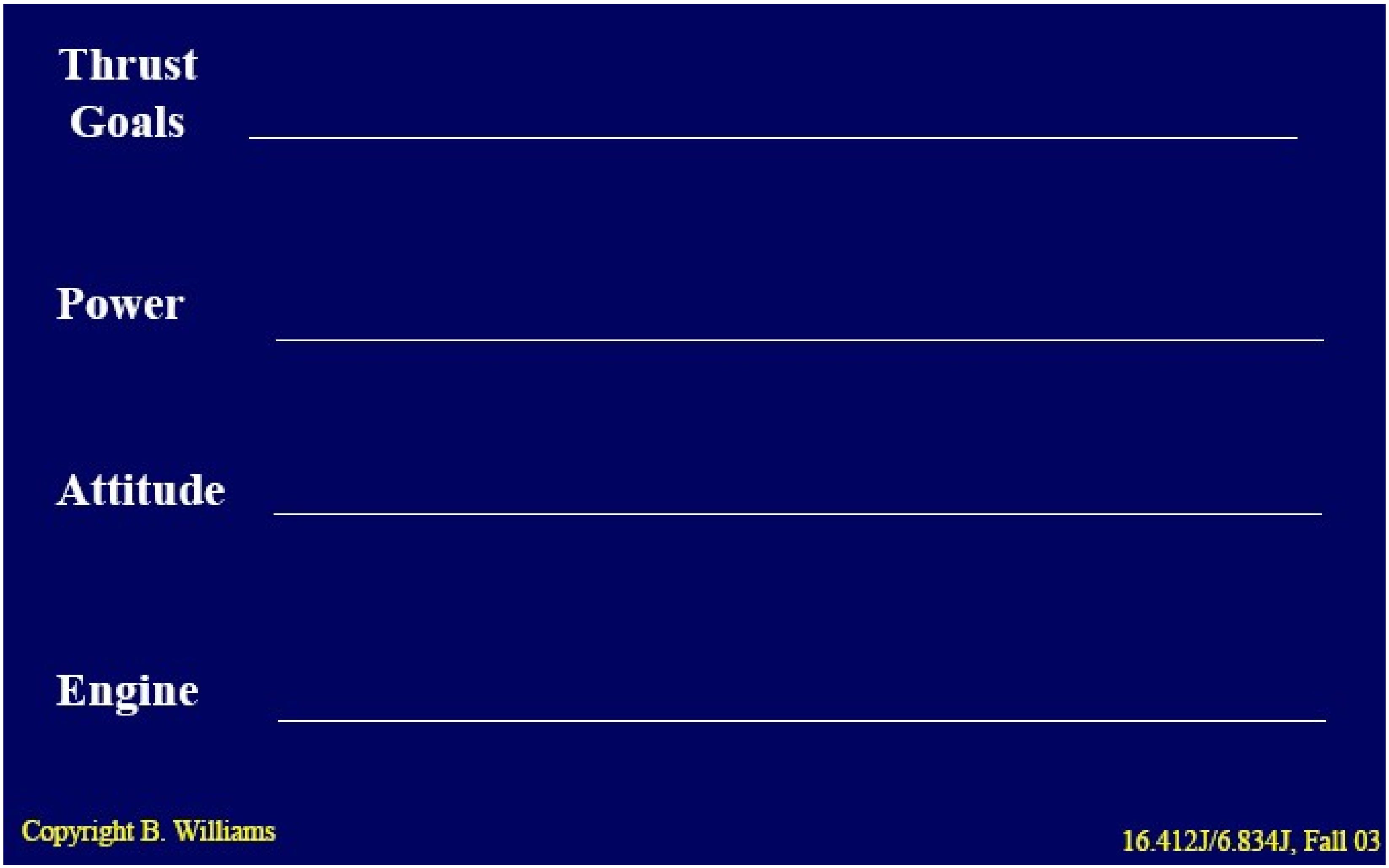
May 21th experiment.

- Diagnose faulty device and
  - Repair by issuing reset.
- Diagnose switch sensor failure.
  - Determine harmless, and continue plan.
- Diagnose thruster stuck closed and
  - Repair by switching to alternate method of thrusting.
- Back to back planning

Remote Agent

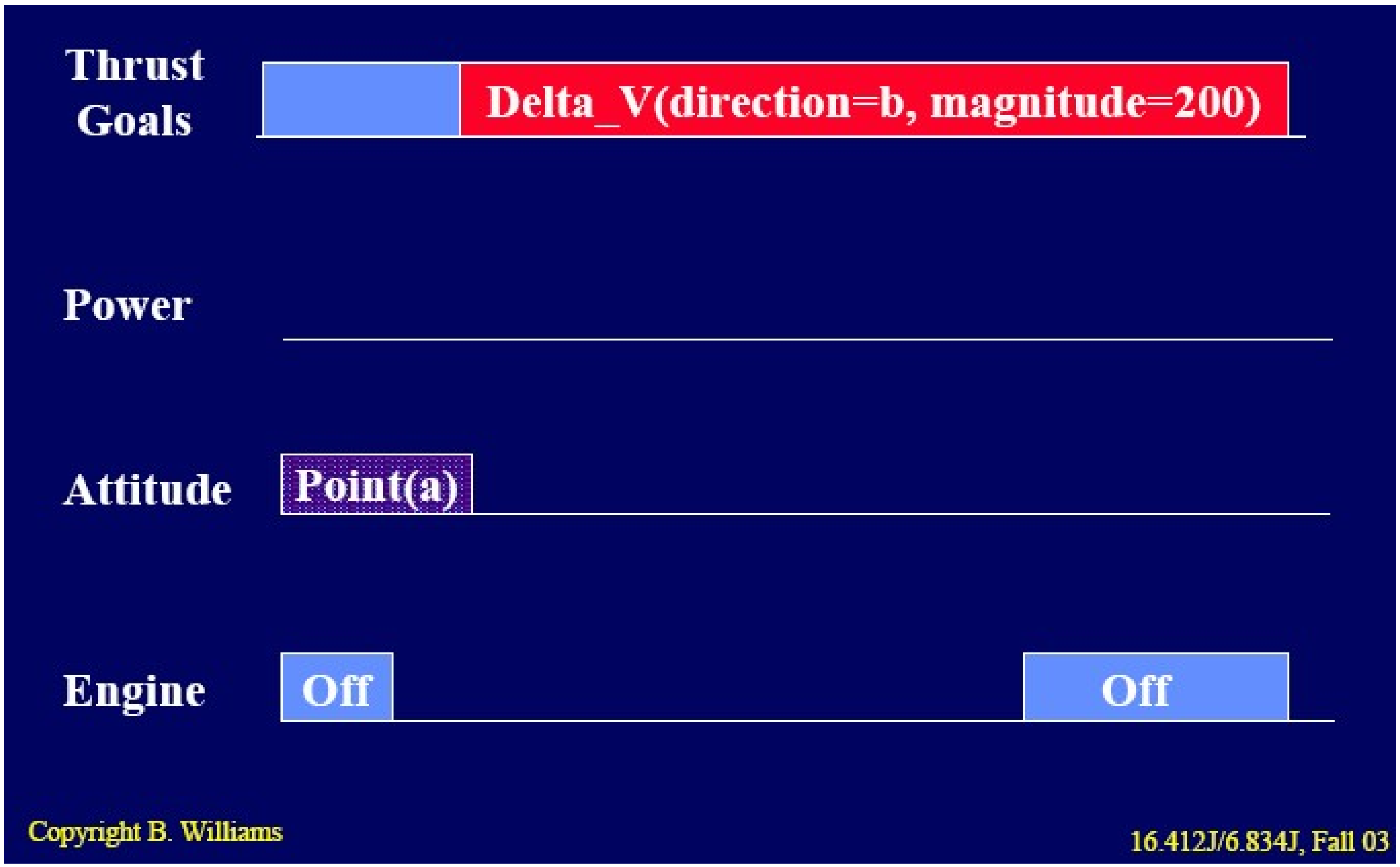


Remote Agent



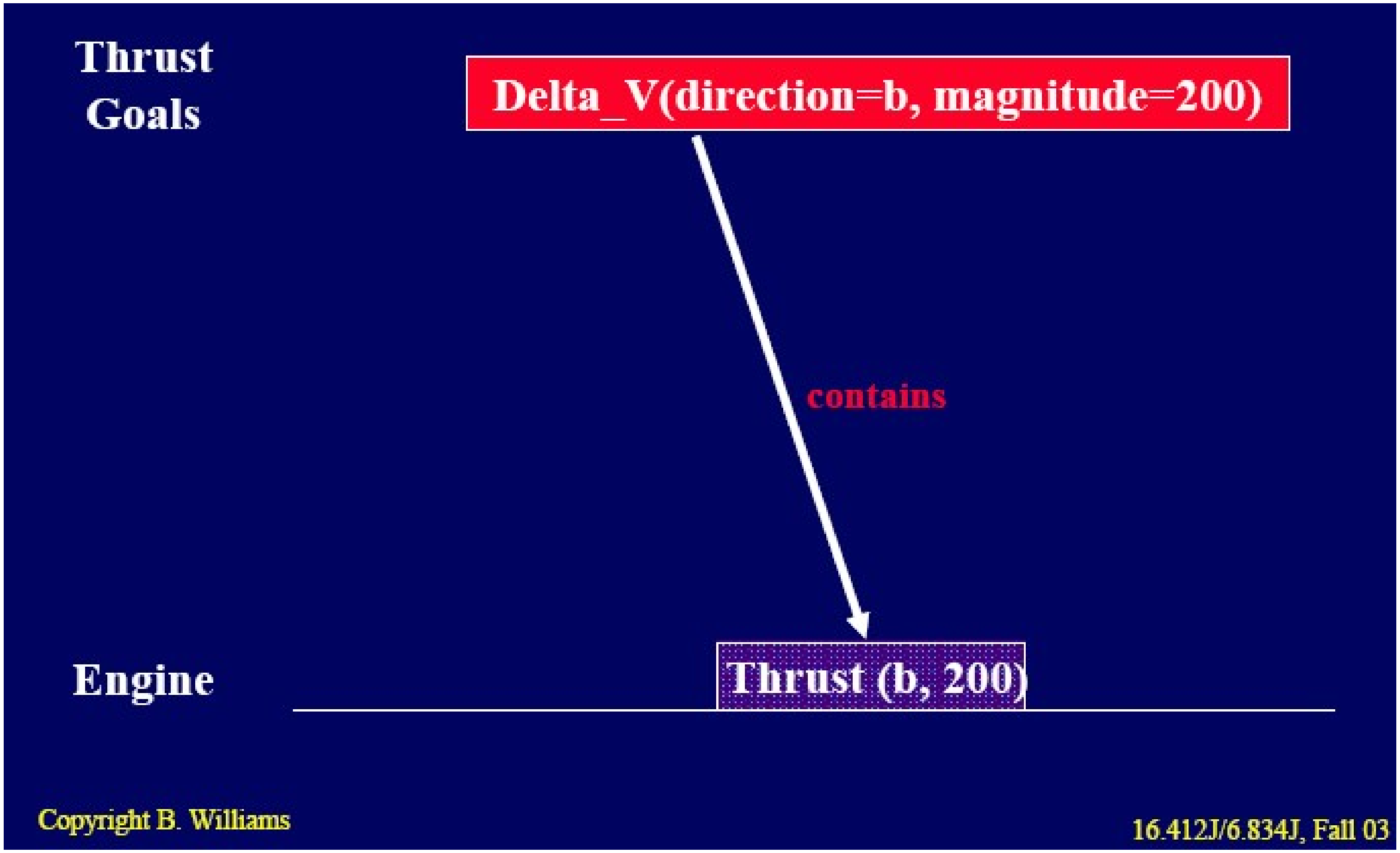
Remote Agent

- Mission Manager



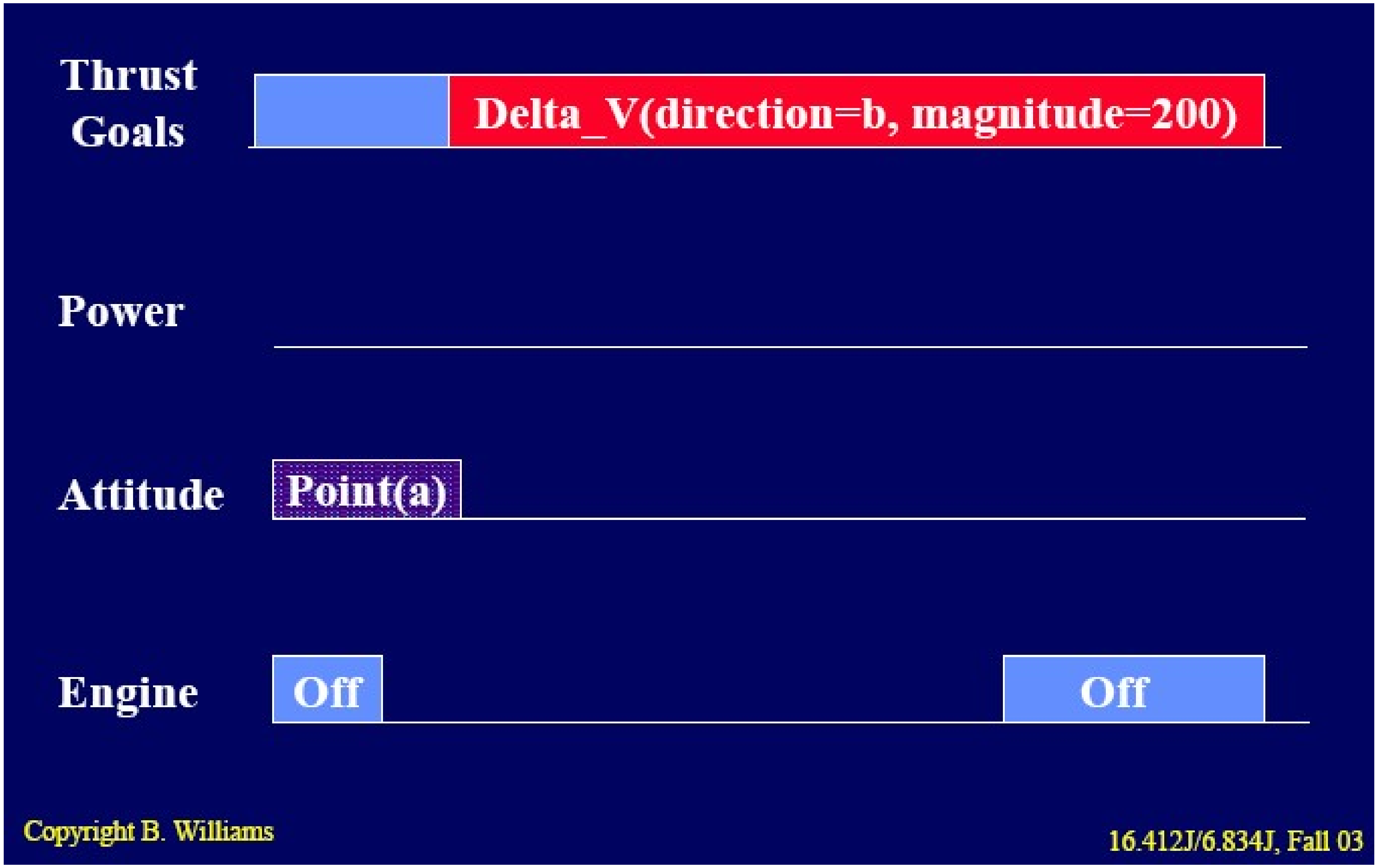
Remote Agent

- Constraints:



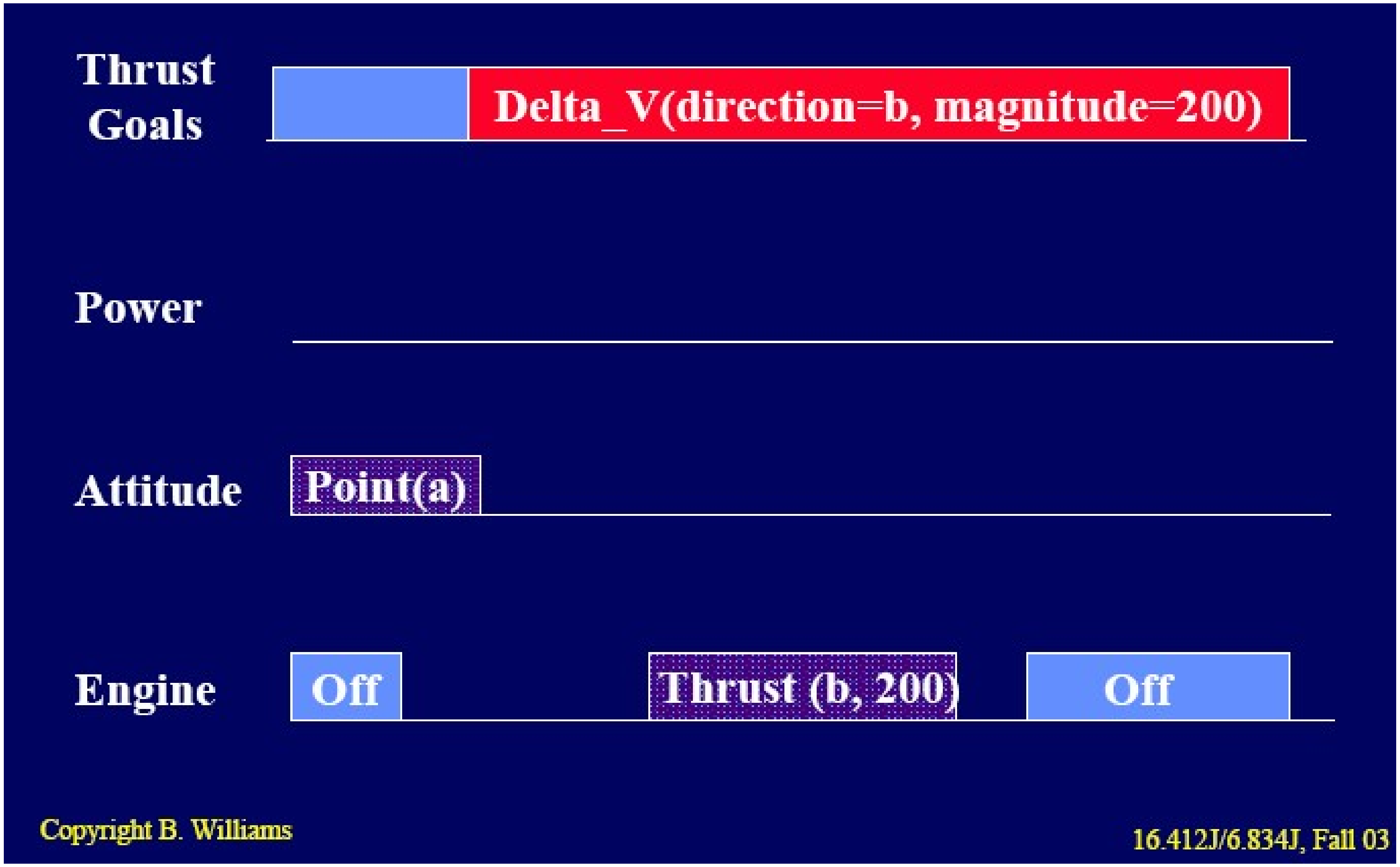
Remote Agent

- Planner starts



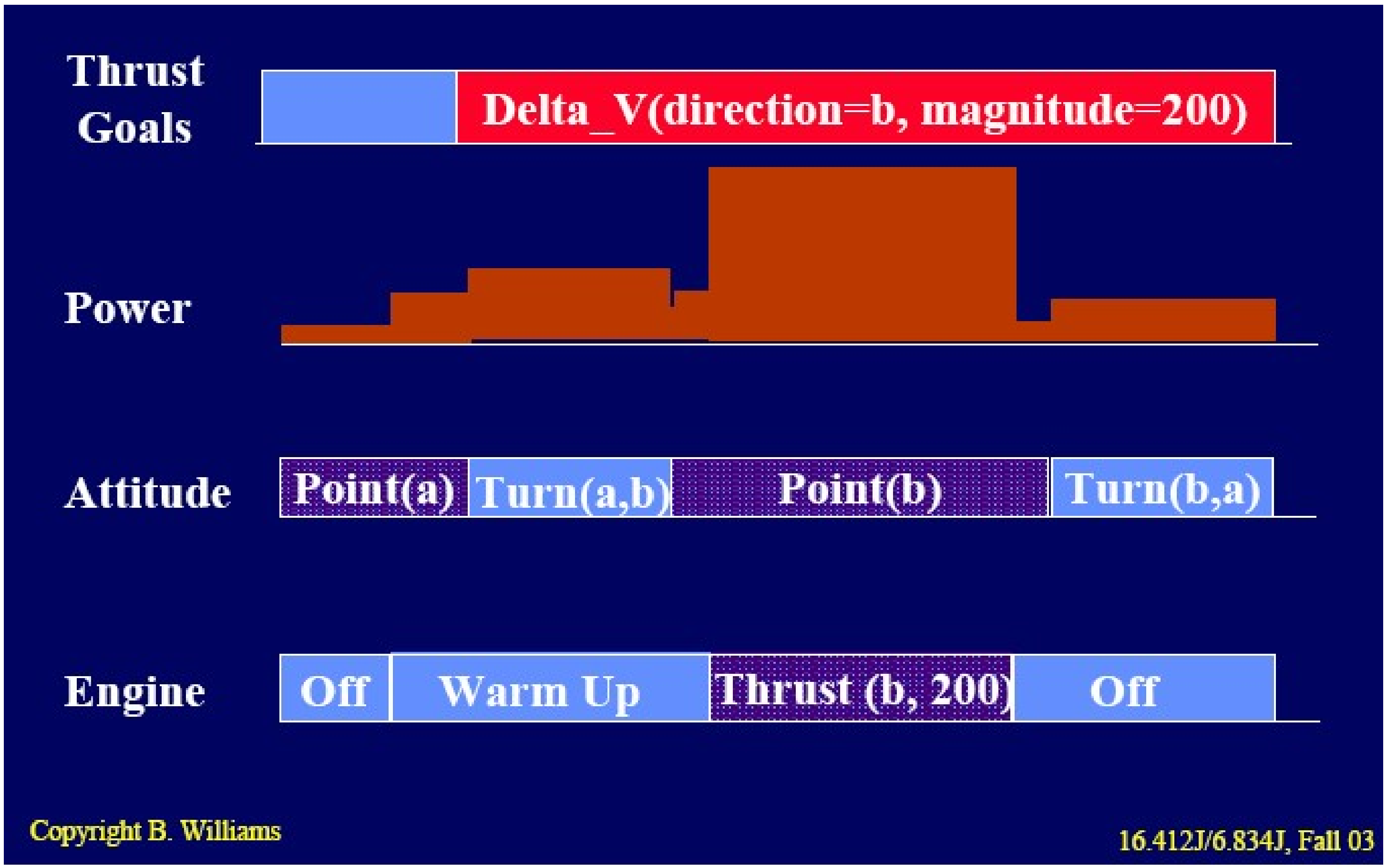
Remote Agent

- Planning



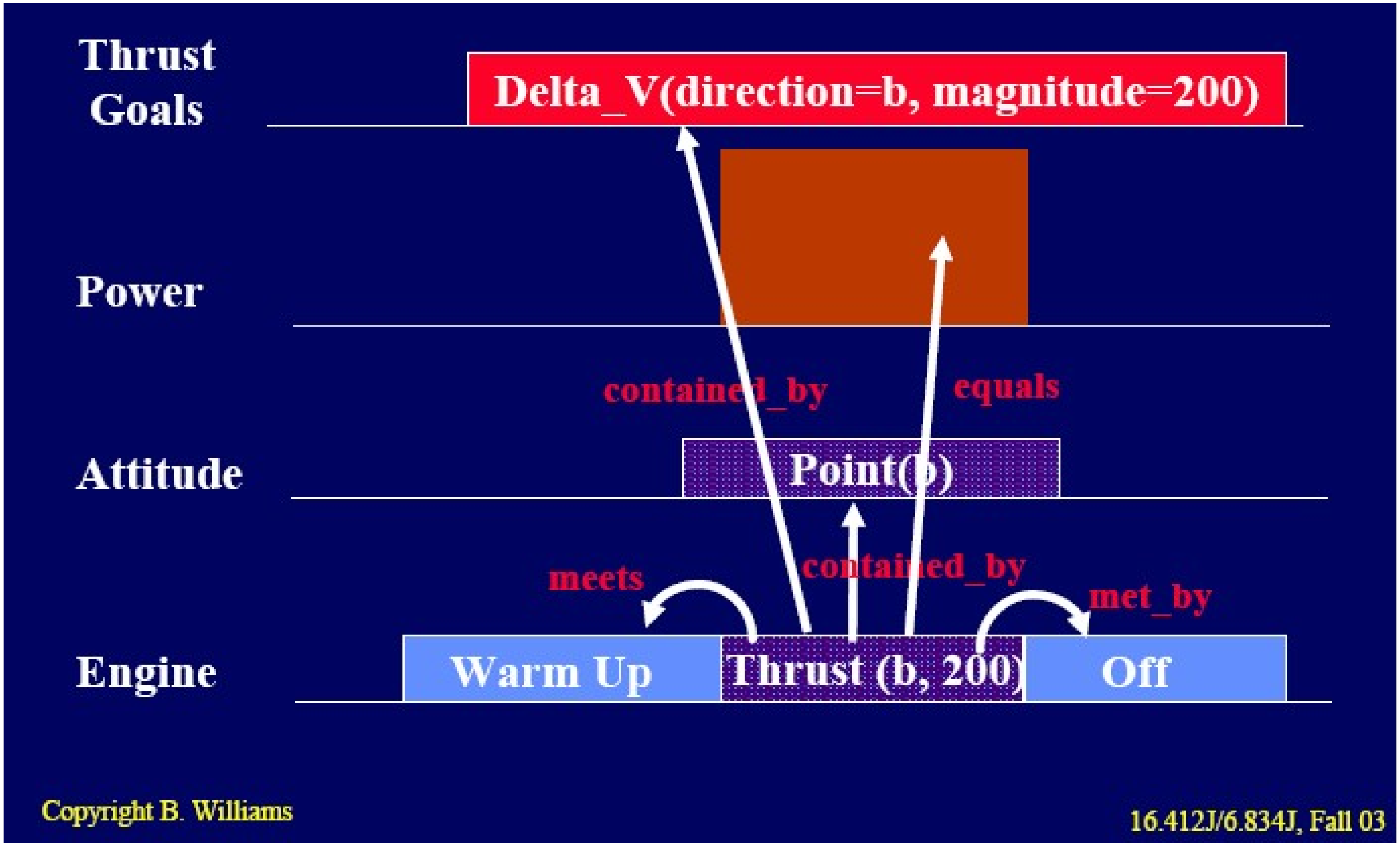
Remote Agent

- Final Plan



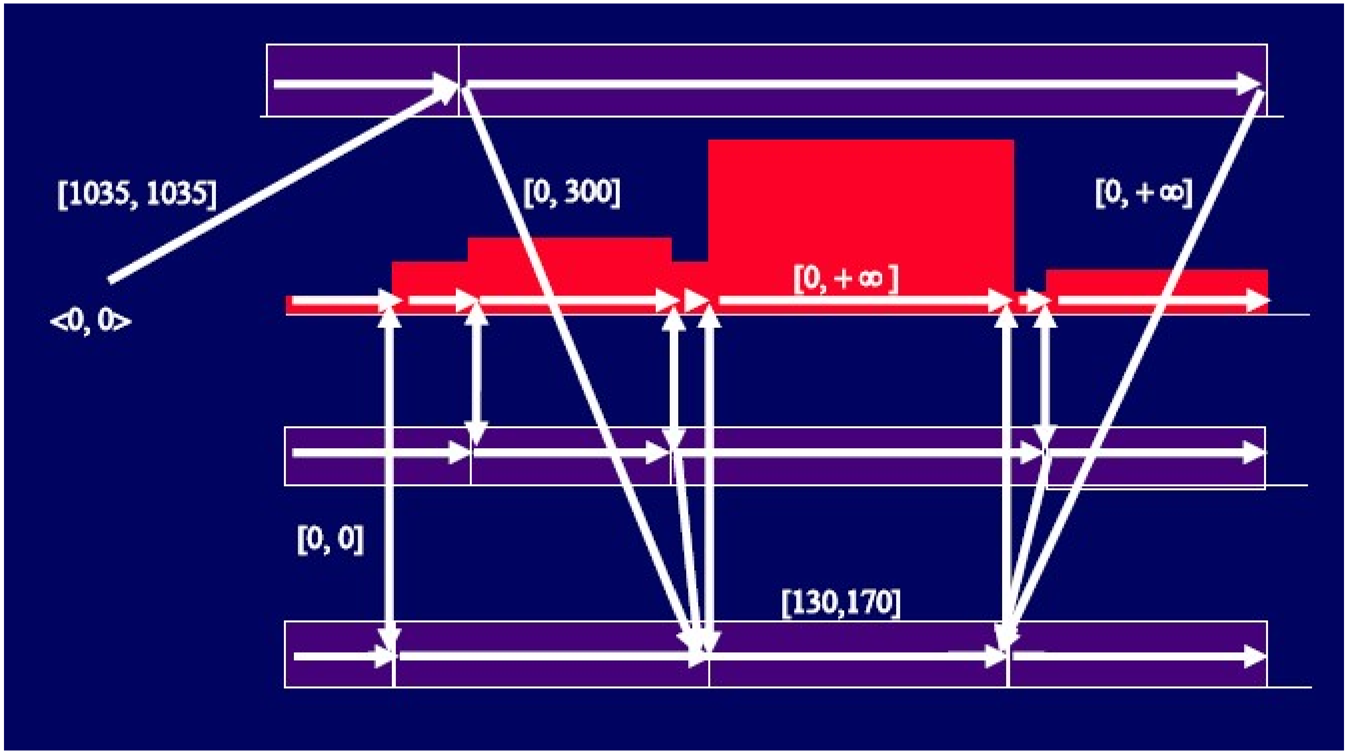
Remote Agent

- Constraints



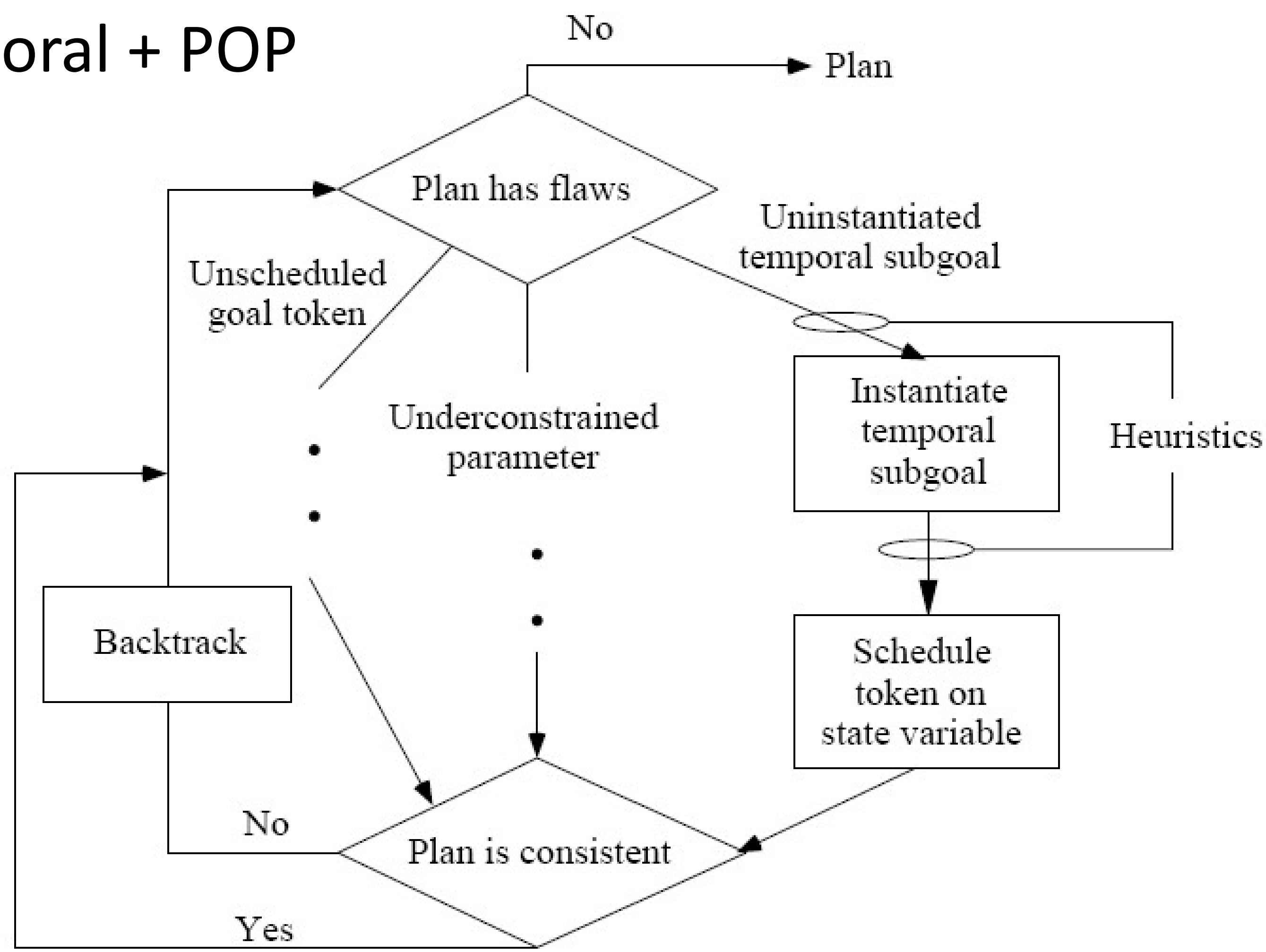
Remote Agent

- Flexible Temporal Plan through least commitment



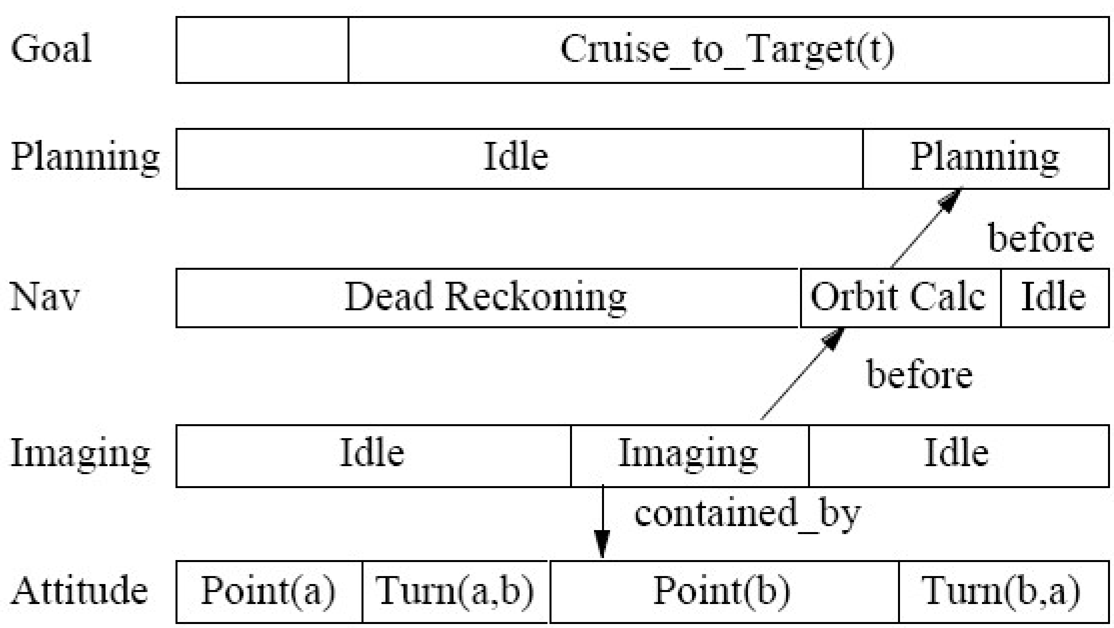
Remote Agent

- Planning
  - Temporal + POP



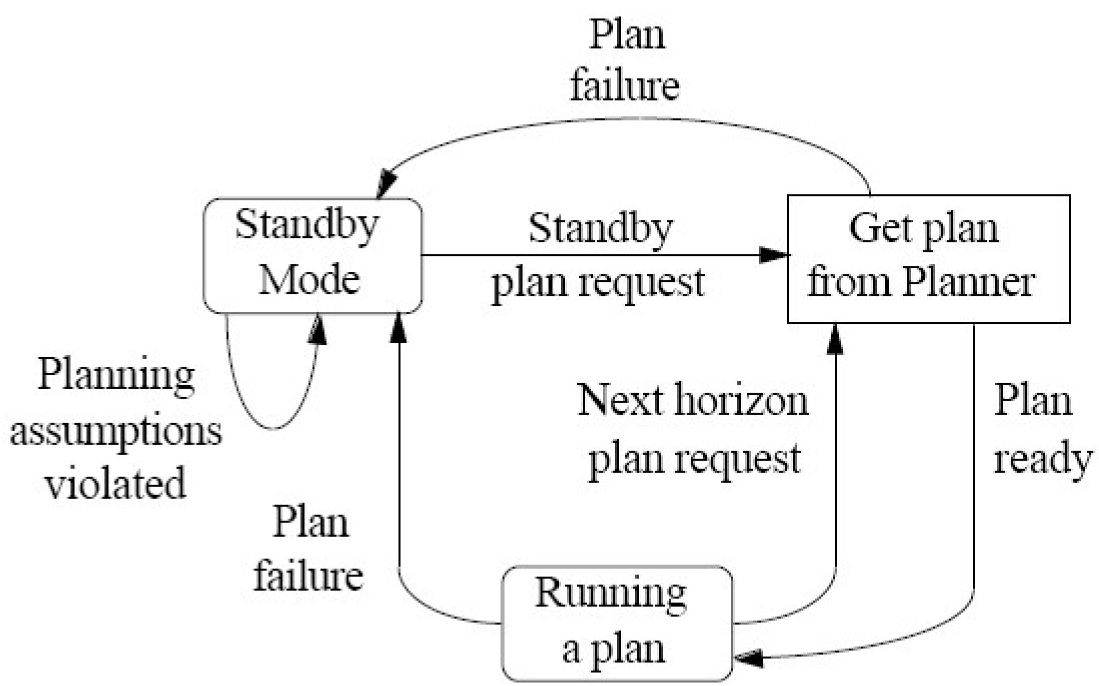
Remote Agent

- Planning to plan



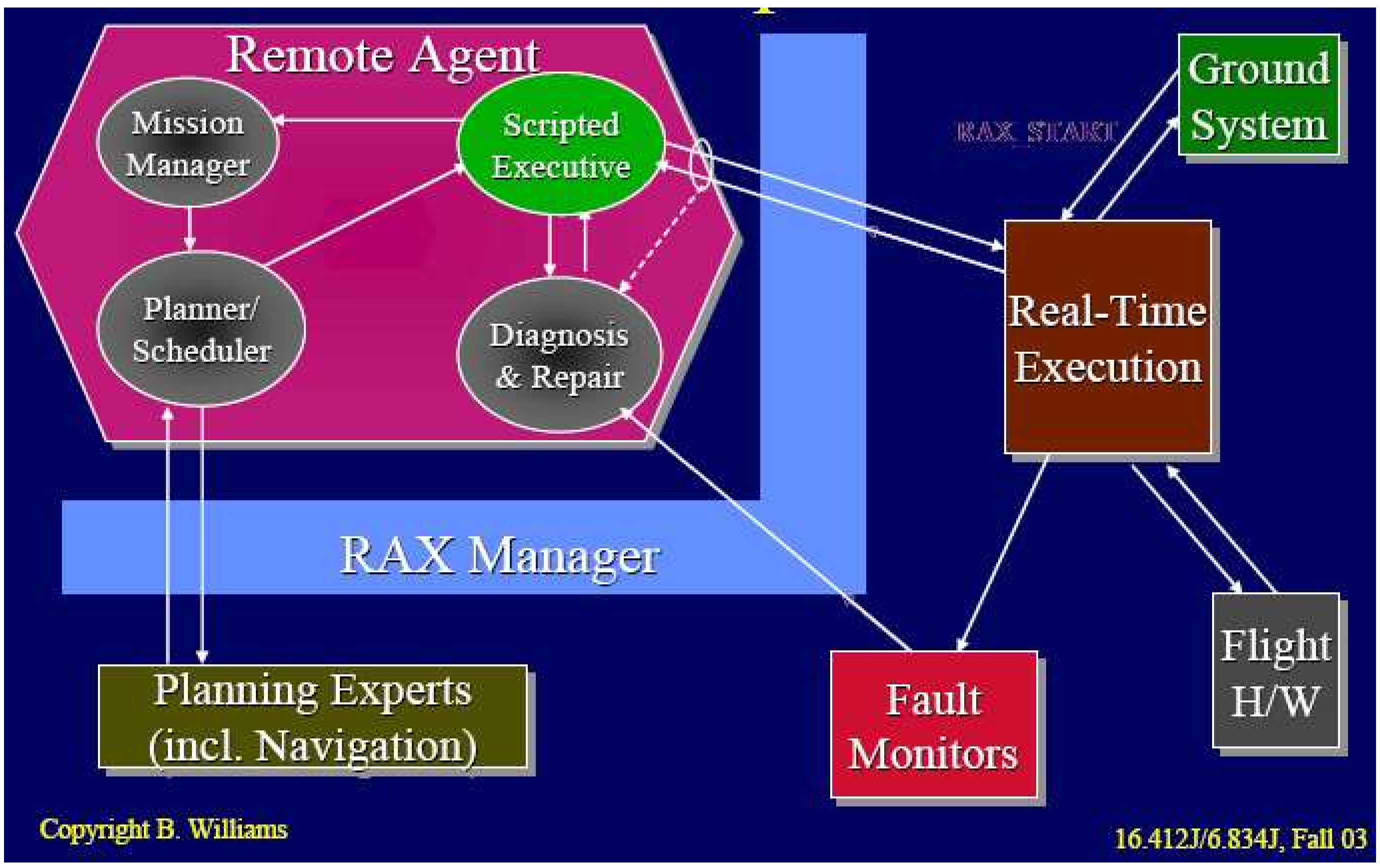
Remote Agent

- Periodic planning and replanning



Remote Agent

- Executive system dispatch tasks

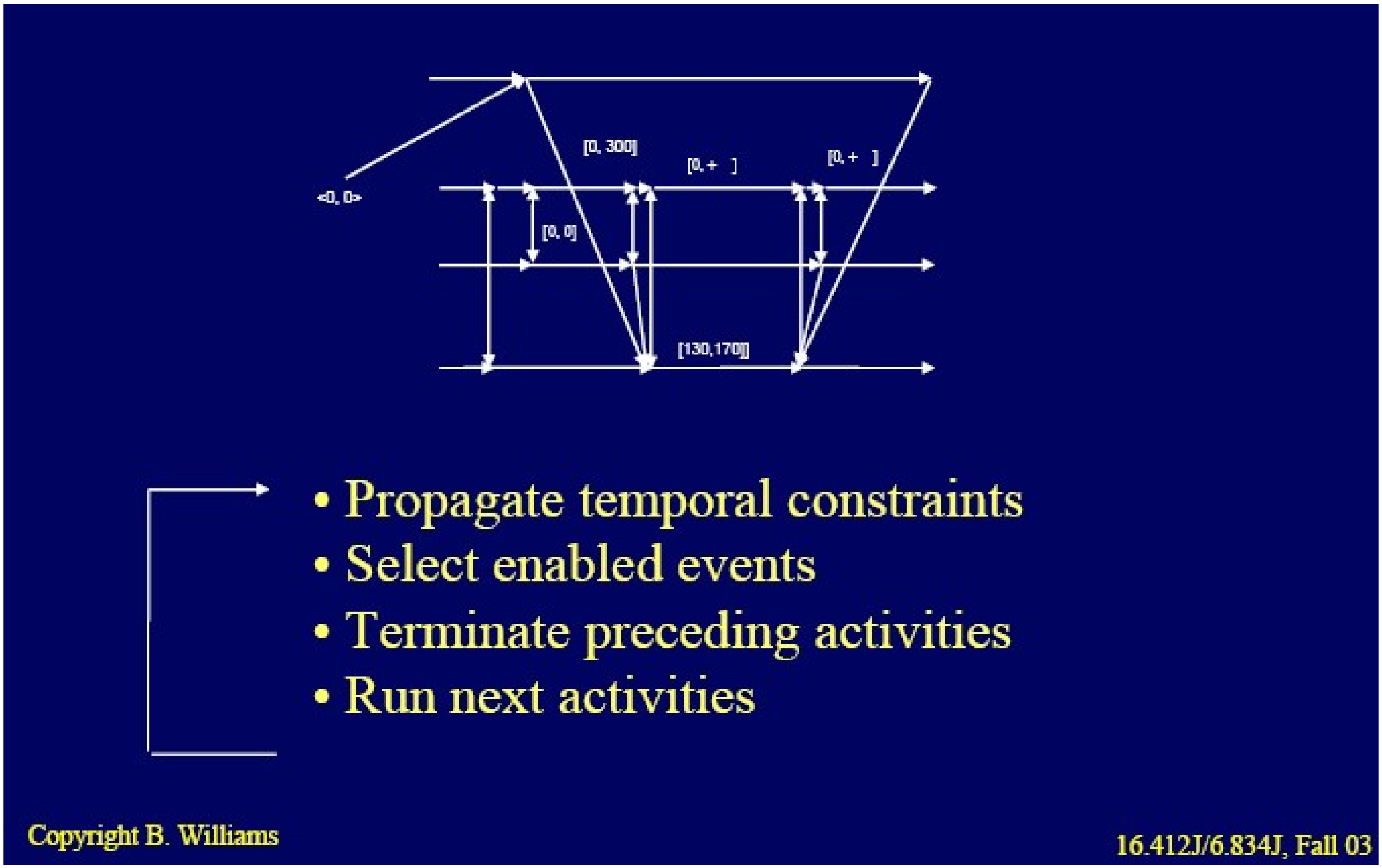


Remote Agent

- The Plan Executor has two duties:
  - Select and Schedule activities for execution
  - Update the network (constraint propagation) after the action execution or execution step (latency)
- Executor Cycle:
  - Activity Graph (STN) from Planner
  - Propagate with latency
  - Enabled time points = scheduled parents (fixed time points)
  - Select and Schedule enabled time points
  - Propagate constraint network given the new binds

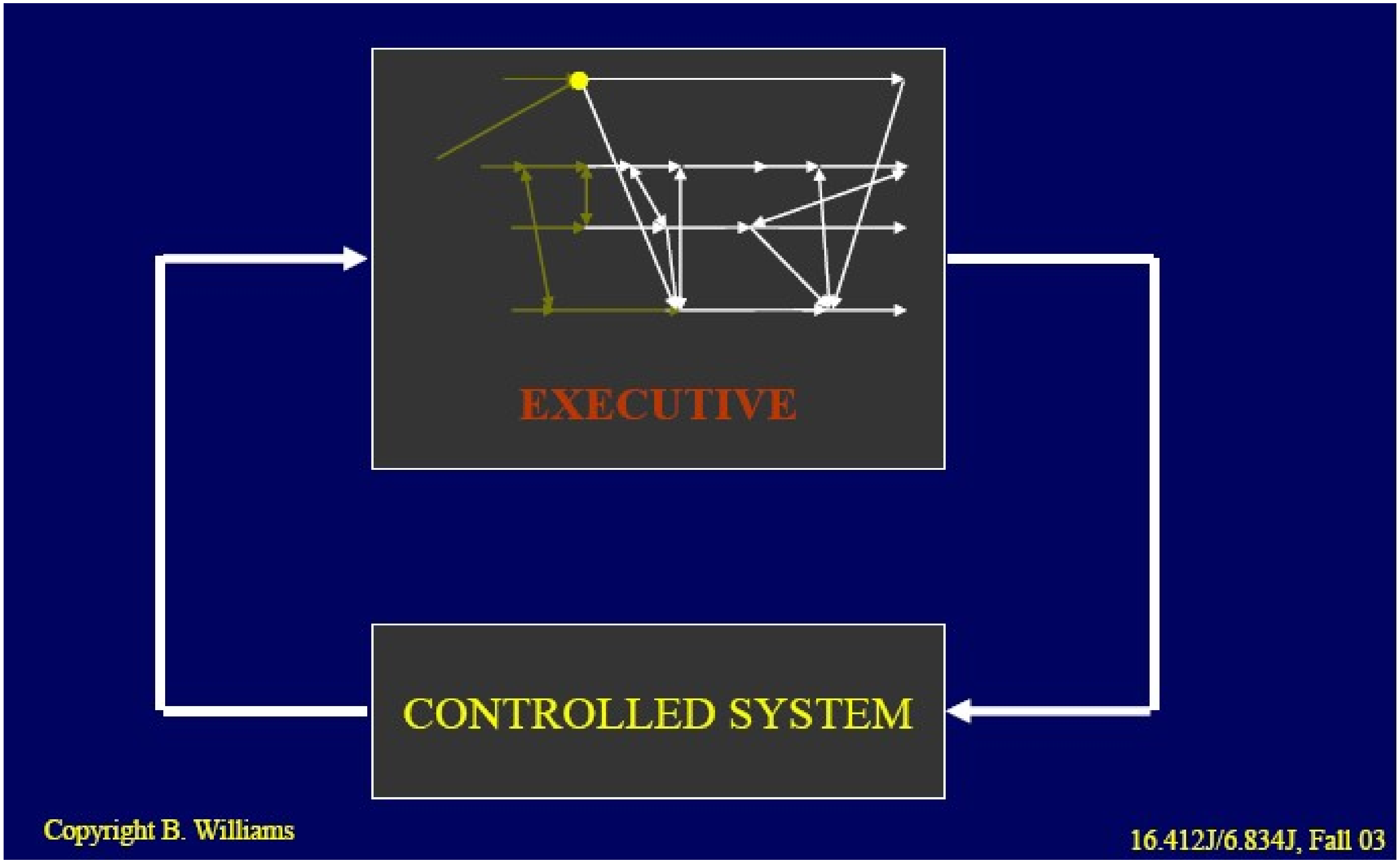
Remote Agent

- Executing Flexible Plans



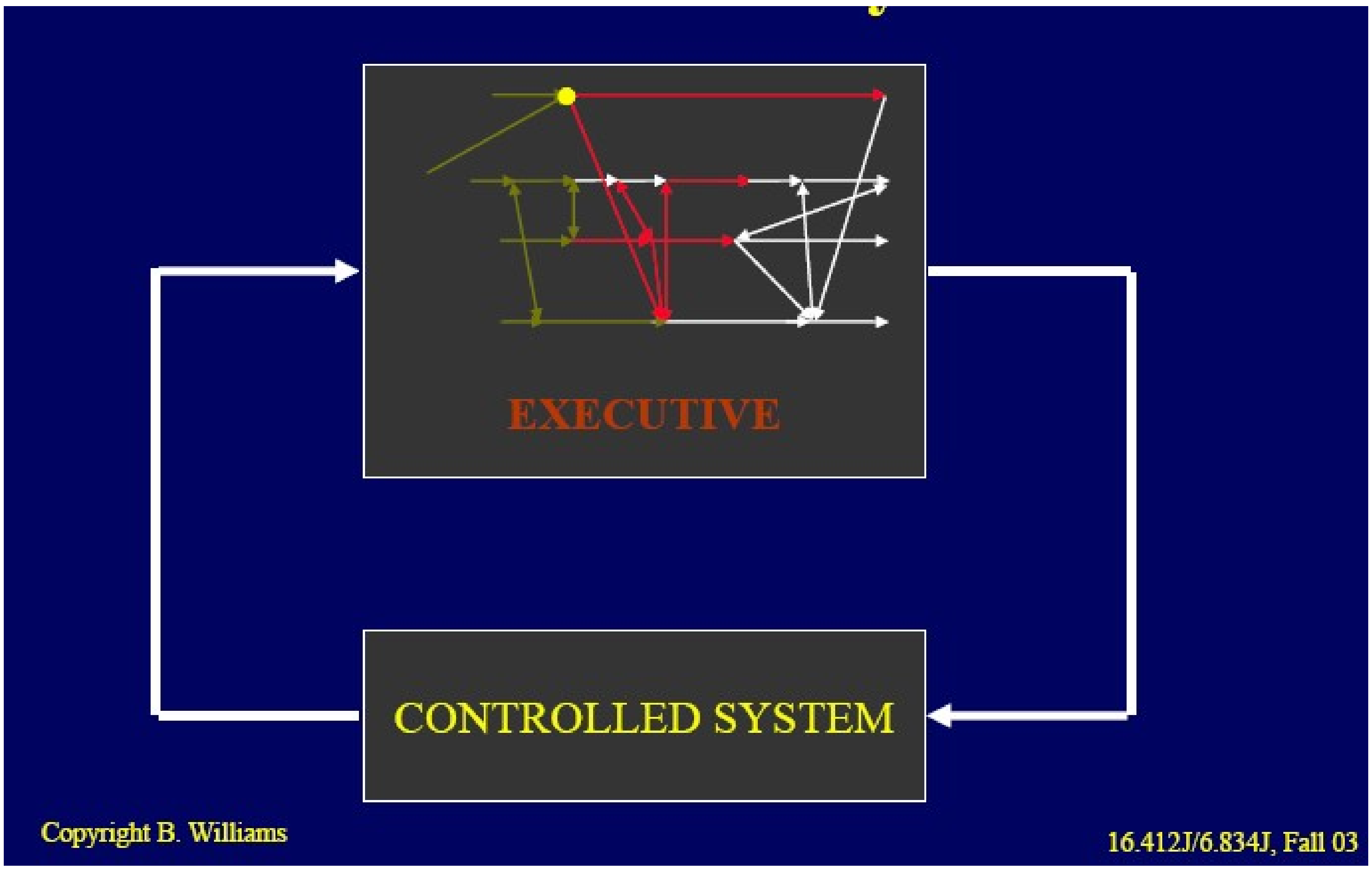
Remote Agent

- Constraint propagation can be costly



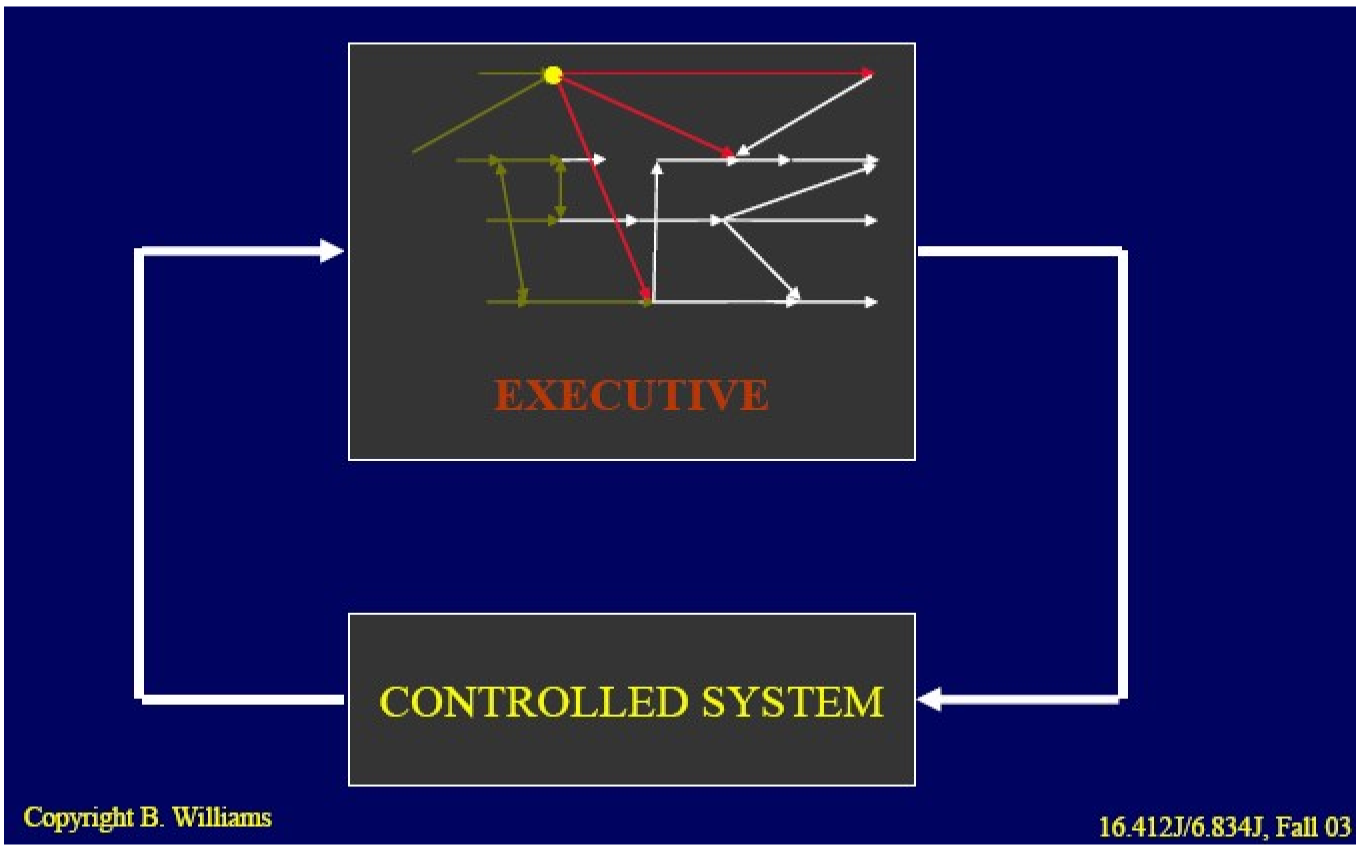
Remote Agent

- Constraint propagation can be costly



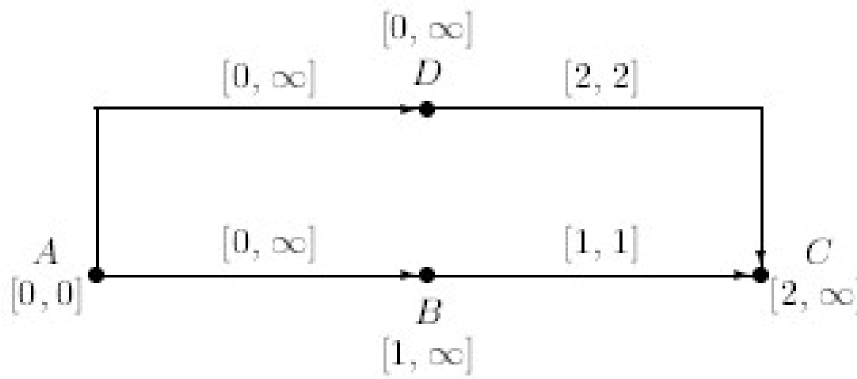
Remote Agent

- Solution: compile temporal constraints to an efficient network

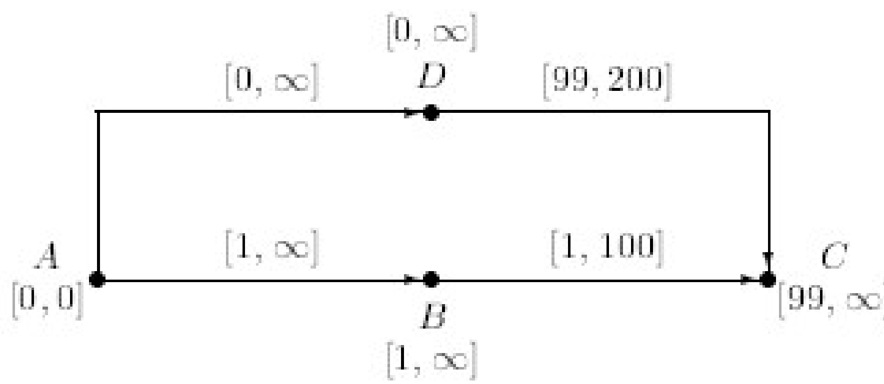


Remote Agent

- <u>Dispatchability</u>
  - Alcuni vincoli non visibili a tempo di esecuzione;

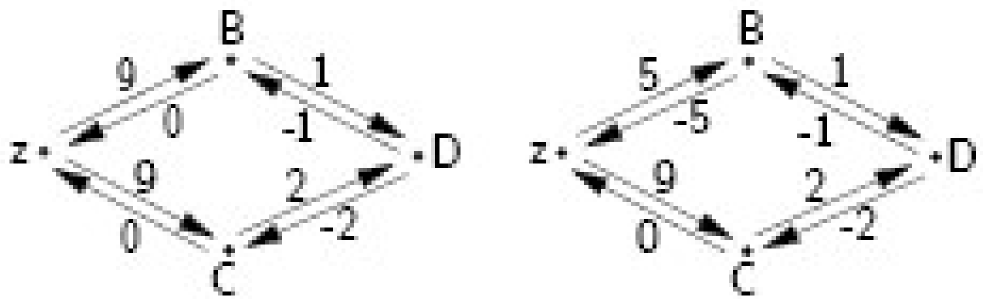


  - Occorre rendere la rete dispatchable aggiungendo vincoli impliciti (e.g. D prima di B)
  - Compilare la rete in forma dispatchable:
    - Introdotti vincoli impliciti
    - Tolti vincoli ridondanti



Dispatchability

A Sample Execution*



After executing B at time 5, it turns out that C must be executed at time 4 (which is already past).

* (Muscettola, Morris, & Tsamardinos 1998)

Dispatcher

Greedy Dispatcher*

While some time-points not yet executed:

- Wait until some time-point is executable.
- If more than one, pick one to execute.
- Propagate updates only to *neighboring* time-points (i.e., do not fully update $\mathcal{D}$).

* (Muscettola, Morris, & Tsamardinos 1998)

Dispatcher

```
TIME DISPATCHING ALGORITHM:
    1. Let
         A = {start_time_point}
         current_time = 0
         S = {}
    2. Arbitrarily pick a time point TP in A such
       that current_time belongs to TP's time bound;
    3. Set TP's execution time to current_time and add
       TP to S;
    4. Propagate the time of execution
       to its IMMEDIATE NEIGHBORS in the distance
       graph;
    5. Put in A all time points TPx such that all
       negative edges starting from TPx have a
       destination that is already in S;
    6. Wait until current_time has advanced to
       some time between
          min{lower_bound(TP) : TP in A}
       and
          min{upper_bound(TP) : TP in A}
    7. Go to 2 until every time point is in S.
```

Dispatchability

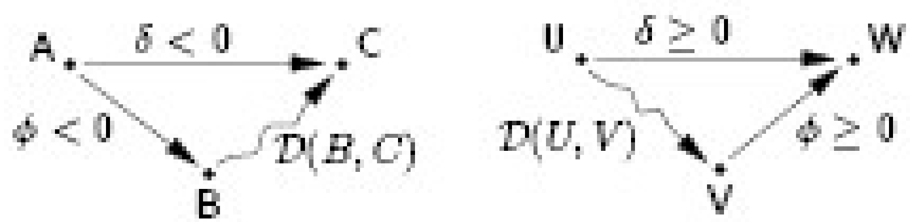
Dispatchability*

- An STN that is guaranteed to be satisfied by the Greedy Dispatcher is called *dispatchable*.
- Any *consistent* STN can be transformed into an equivalent *dispatchable* STN.
- Step I: The corresponding All-Pairs graph is equivalent and dispatchable.
- Step II: Remove *lower- and upper-dominated* edges (does not affect dispatchability).

* (Muscettola, Morris, & Tsamardinos 1998).

Dispatchability

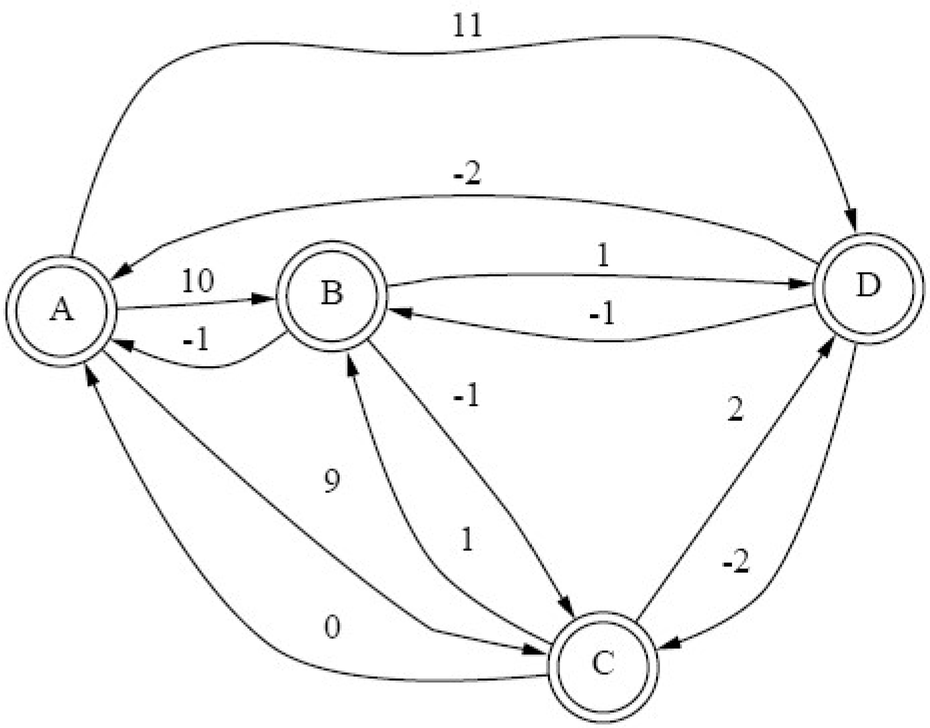
Lower and Upper Dominance*



- The *negative edge* AC is *lower-dominated* if:
  $\delta = \phi + \mathcal{D}(B, C)$.
- The *non-negative edge* UW is *upper-domin'd* if:
  $\delta = \mathcal{D}(U, V) + \phi$.

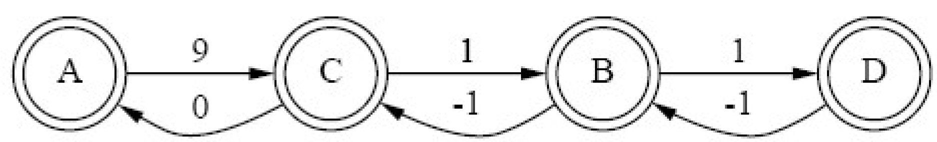
* (Muscettola, Morris, & Tsamardinos 1998)

Dispatchability



All pair graph

Filtered graph



Controllability

- Alcune attività non sono controllabili, ma solo osservabili
- E.g. after start_turn, end_turn ? Quando finisce?
- Il grafo delle attività STN contiene time point controllabili e non controllabili
- Le attività non controllabili non possono essere schedulate, ma solo osservate
- Propagazione?

Controllability

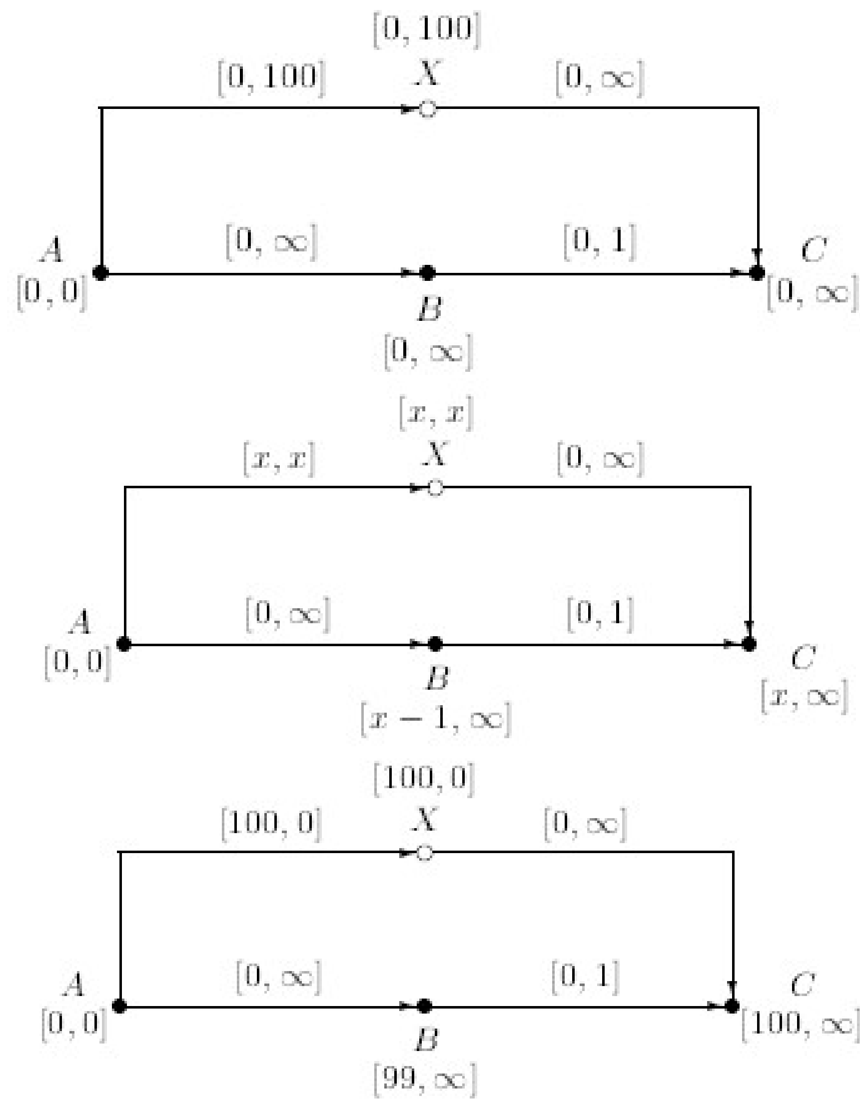
Controllability Issues*

- In real-world applications, an agent may only control some time-points directly; others may be controlled by other agents or Nature.
- Such a network is called *controllable* if there exists a strategy for the agent to execute the time-points under its direct control that will ensure the consistency of the network—no matter how the other agents or Nature execute their time-points.

* (Vidal & Ghallab 1995; Vidal & Fargier)

Controllability

- Gestire eventi non controllabili
- Es. Se B schedulato prima di X, B vincola X
  - Soluzione Dinamica: B dopo X
  - Soluzione Forte: B a 99



Controllability

- **Weak Controllability**:
  - For each uncontrollable event there exists a scheduling for the execution;
- **Strong Controllability**:
  - There exists a scheduling that works for all the uncontrollable events;
- **Dynamic Controllability**:
  - For each uncontrollable past event there exists a scheduling for the execution.