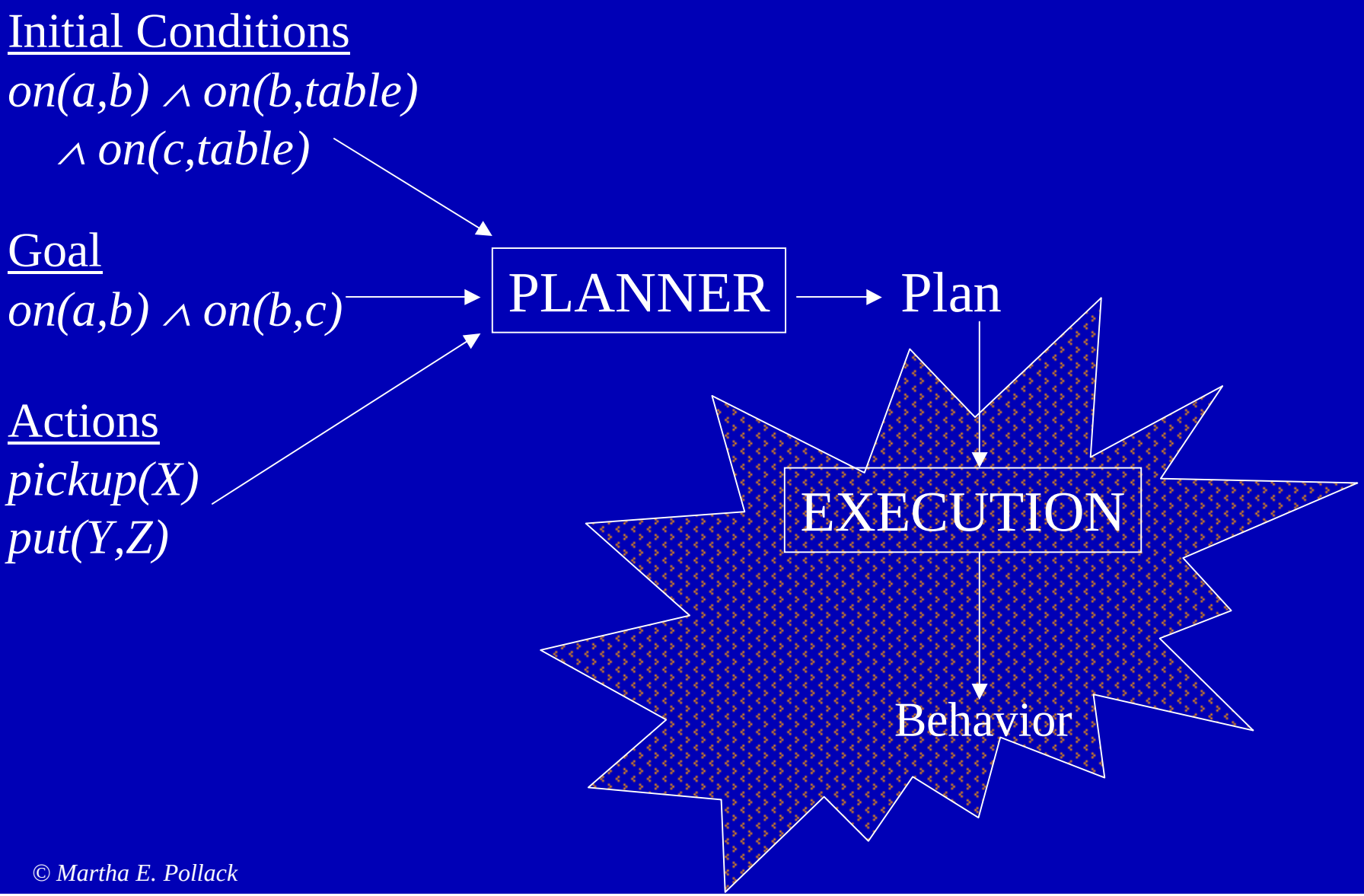
PLANET International Summer School
On AI Planning 2002

Planning and Execution

Martha E. Pollack
University of Michigan
www.eecs.umich.edu/~pollackm

Classical Planning



Classical Planning Assumptions

- World is static (and therefore single agent).
- Actions are deterministic.
- Planning agent is omniscient.
- All goals are known at the outset.
- *Consequently*, everything will "go as planned": execution is a trivial matter.

Planning in "Real Life"

- Triage in the hospital emergency room
- Initial State(s): Sick or injured patients
- Goal State(s): Each patient treated appropriately
- Actions: Perform medical tests, administer medication, suture wounds, apply casts to broken bones, send to surgery. . .

Typical Plan (Protocol)

- US NINDS Guidelines for Treatment of Potential Stroke (Thrombolytic) Patient

ACTION	TIME TARGET
Hospital door to doctor	10 minutes
Door to neurological expert	15 minutes
Door to CT scan completion	25 minutes
Door to CT scan interpretation	45 minutes
Depending on test results, door to treatment	60 minutes
Depending on test results, admission to monitored bed	3 hours

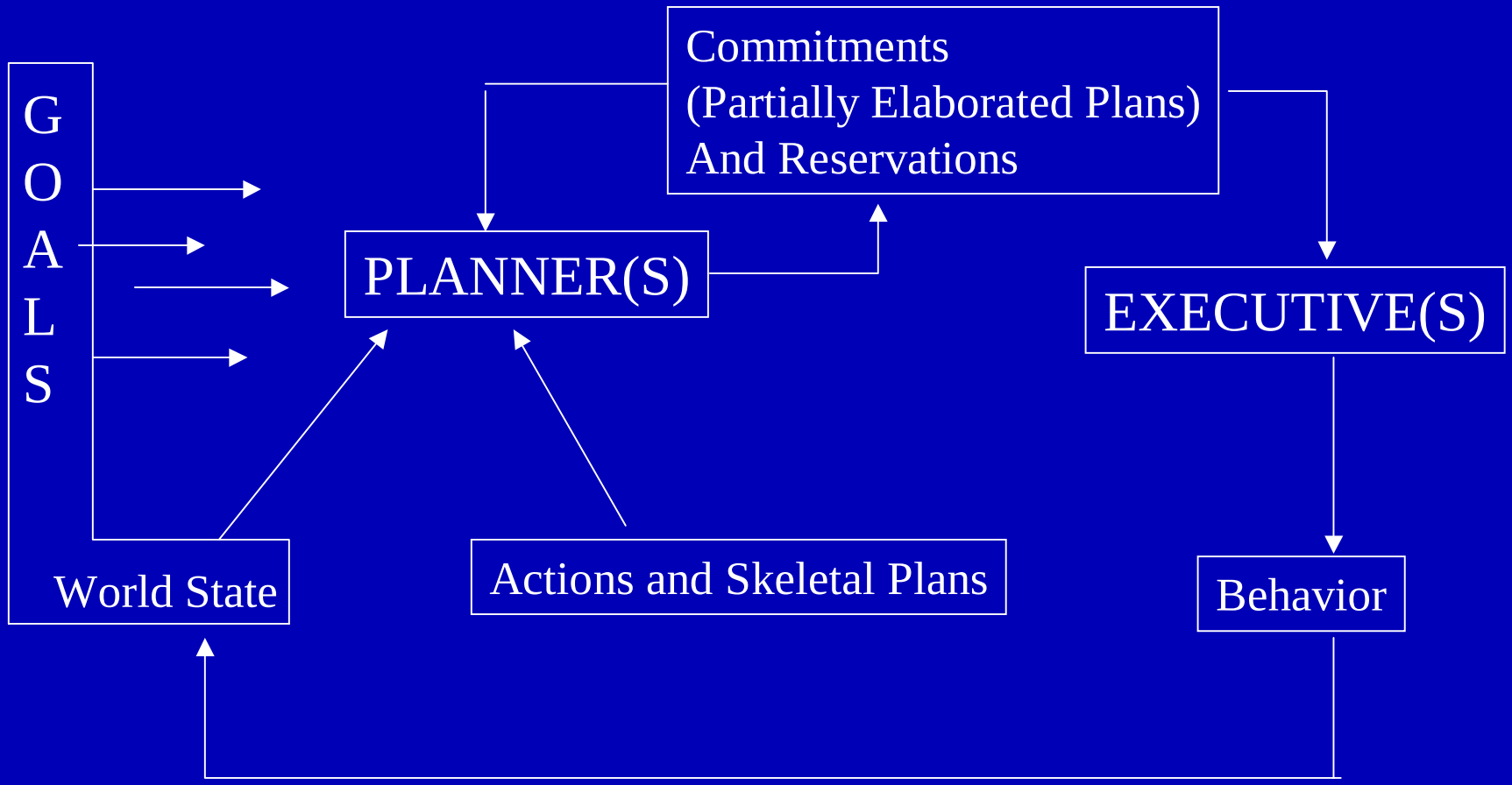
Problem Characteristics

- World is dynamic and multi-agent
  - E.g., many doctors and nurses
- Actions are not deterministic
  - E.g., patients respond differently to the same treatment
- Planning and executing agents are not omniscient
  - E.g., physicians don't know each patient's medical history
- Planning "problems" (i.e., patients) arrive asynchronously

$\Rightarrow$ Planning and Execution need to be interleaved and actively managed

- Skeletal plans (protocols) are known in advance

Integrated Model of Planning and Execution



Domains

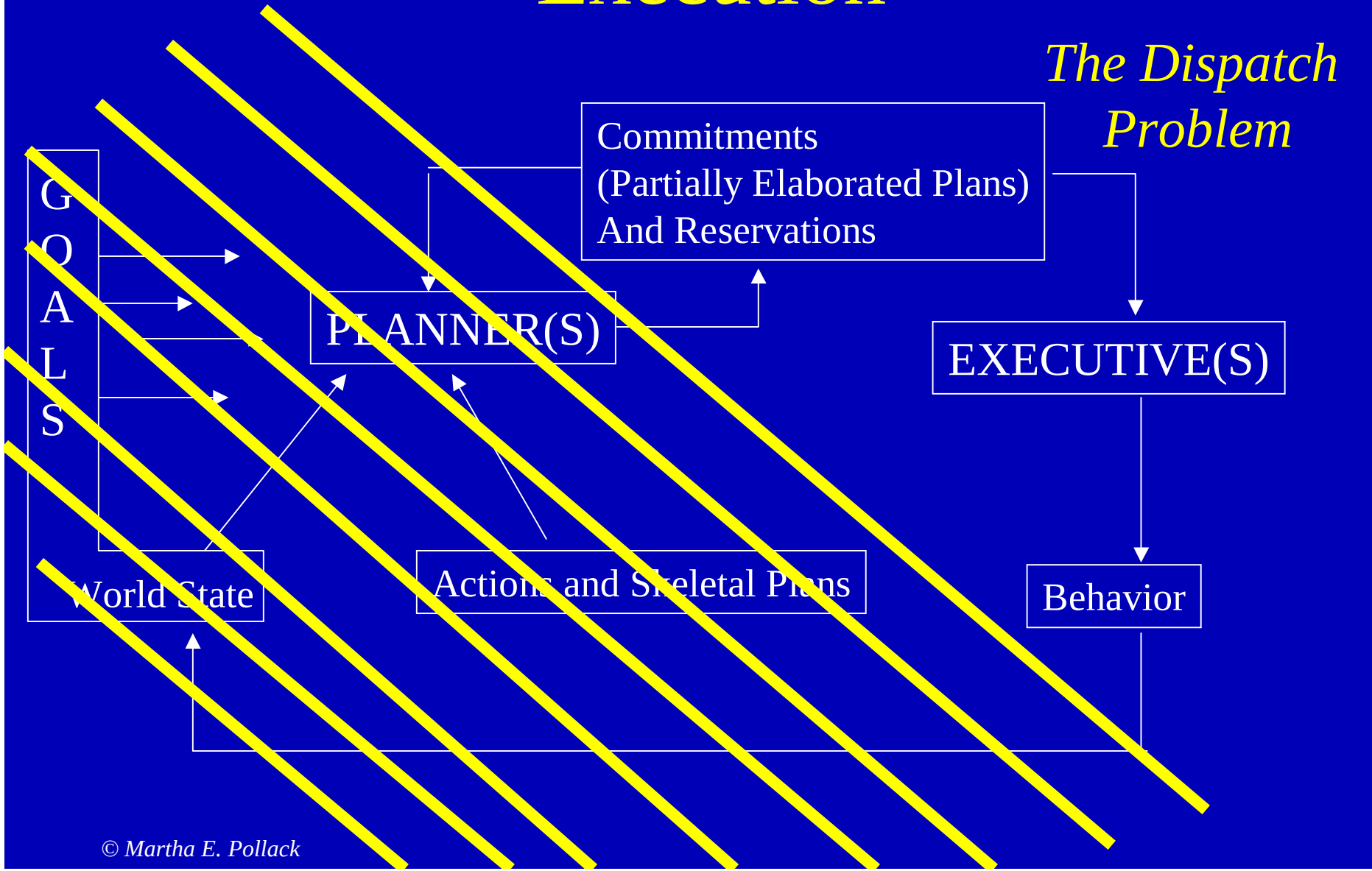
- AI systems for planning and execution have been developed for
  - Automated spacecraft
  - Mobile robot applications
  - Military campaign management
  - Air traffic control
  - Cognitive orthotics
  - and many more applications

Outline

Lecture #1: Execution

Lecture #2: Planning and Execution

Integrated Model of Planning and Execution



Today's Outline

√ 1. Introduction

2. Satisfying Temporal Constraints during Execution

   A. Temporal Constraint Networks: Simple Temporal Networks

   B. Execution Management (Plan Dispatch) for STPs

   C. More Expressive Networks: Disjunctive Temporal Networks

   D. Plan Dispatch *and Plan Update* for DTPs

   E. Yet More Expressive Networks

Temporal Constraints in Plans

- Simplest Case: precise information about the timing and duration of all events
  - Then dispatch is easy.
- In general:
  - Such information may be unknown and/or
  - such an encoding may be undesirable (inflexible), so
  - prefer to provide intervals on times of occurrence

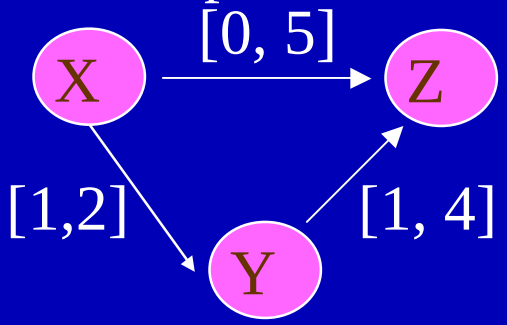
Temporal Constraint Problems

- Family of constraint-satisfaction problems (CSPs), $<V,E>$ where
  - V = events
  - E = interval-based constraints
- Members of the family are defined by the form of the constraints

Simple Temporal Problems

- A Simple Temporal Problem (STP) is a constraint-satisfaction problem <V,E> such that V is a set of real-valued temporal variables and E is a set of constraints of the form $y - x \leq u$, where $u \in R$.
  - W.l.o.g. assume $u \in Z$

Example STN



Constraints:

$1 \leq y - x \leq 2$

$0 \leq z - x \leq 5$

$1 \leq z - y \leq 4$

One Solution:

$\{x = 0, y = 1, z = 4\}$

$1 \leq y - x \leq 2 \Leftrightarrow y - x \leq 2 \wedge x - y \leq -1$

STP Example

Model temporal plans by associating nodes with the start and end points of each action

EVENT	NAME
Patient enters hospital	E
Start(Patient seen by doctor)	D_S
End(Patient seen by doctor)	D_E
Start(Patient seen by neurologist)	N_S
End(Patient seen by neurologist)	N_E
Start(CT scan)	C_S
End(CT scan)	C_E
Start(CT scan interpretation)	I_S
End(CT scan interpretation)	I_E
Start(Treatment)	T_S
Start(Admission to monitored bed)	A_S

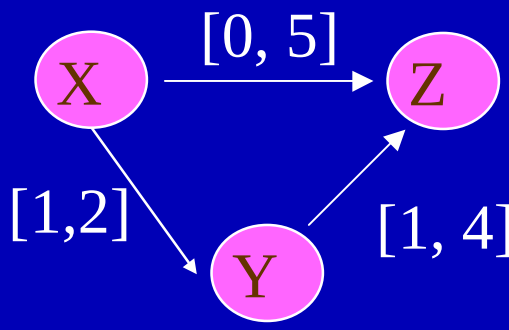
STP Example

CONSTRAINT	SEMANTICS
$D_S - E \leq 10$	Door to doctor by 10 minutes
$N_S - E \leq 15$	Access to neurologist by 15 minutes
$C_E - E \leq 25$	Door to CT scan completion by 25 minutes
$I_E - E \leq 45$	Door to CT scan interpretation by 45 minutes
$T_S - E \leq 60$	Door to treatment by 60 minutes
$A_S - E \leq 180$	Admission to monitored bed within 3 hours
$C_E - I_S \leq 0$	Scan interpretation starts after scan completion
$I_E - T_S \leq 0$	Treatment begins after scan interpretation

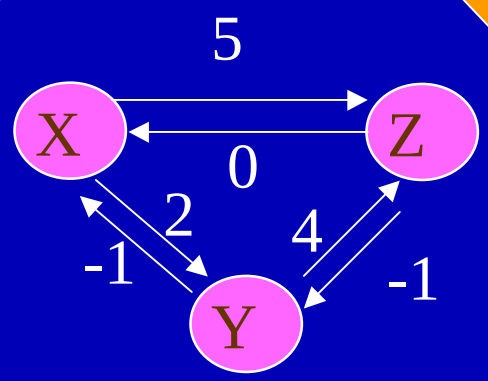
Solving STPs

- An STP is *consistent* (has a solution) if its distance graph contains no negative cycles.
- Can determine this in polynomial time, using all-pairs shortest path algorithms (e.g., Floyd-Warshall)
- If consistent, can directly "read off" one solution from the shortest path matrix—but typically there will be many others as well.

An Example

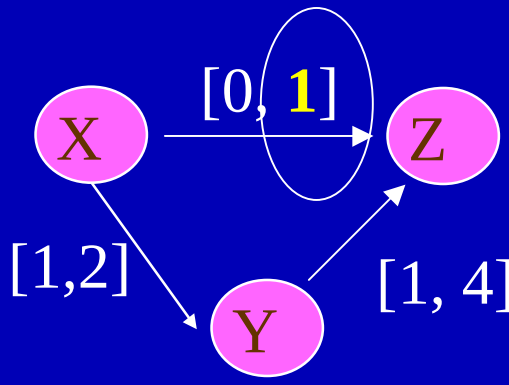


	X	Y	Z
X	**0**	2	5
Y	-1	**0**	4
Z	-2	-1	**0**

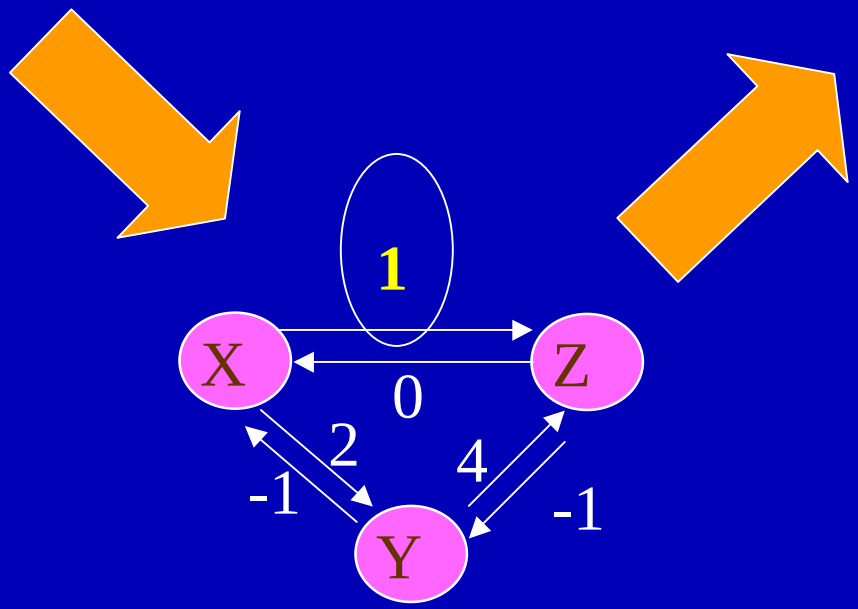


One Solution:
{x = 0,y = 2, z = 5}
An alternative:
{x = 0, y = 1, z = 2}

Another Example

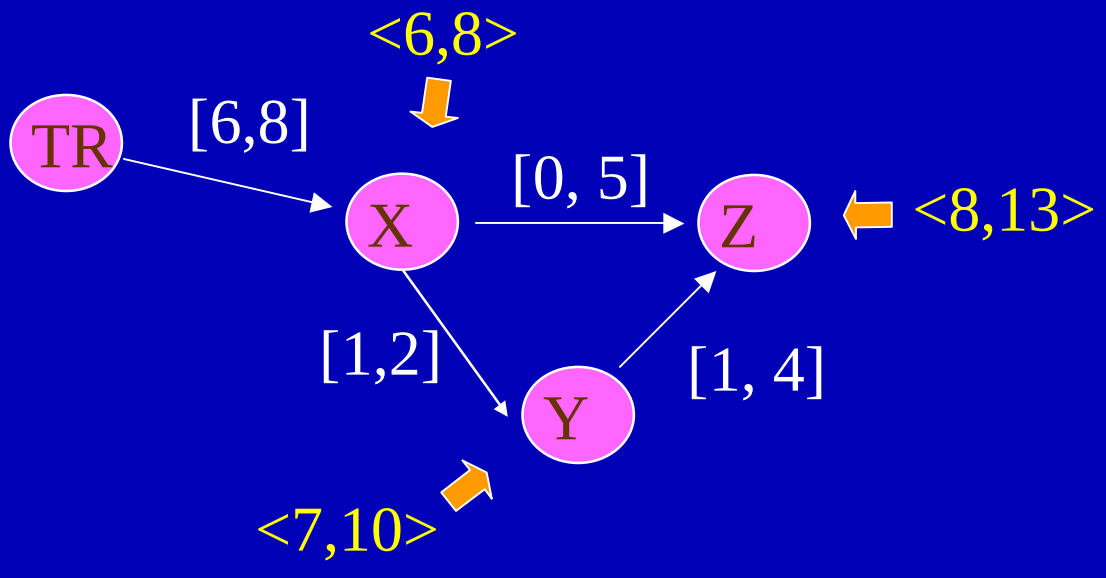


	X	Y	Z
X	**-1**		
Y			
Z			



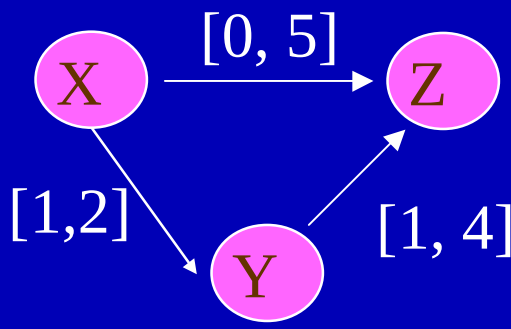
TR's and TW's

- Use a *Temporal Reference Point (TR)* to specify absolute clock times
- Compute the *Time Window (TW)* for every event *e*
  - Minimal distance to/from *TR*



Decomposability

- An STP is *decomposable* if every locally consistent assignment can be extended to a solution.



X = 0

Z = 0, satisfying Con({X,Z})

No way to extend with an assignment to Y –
not decomposable

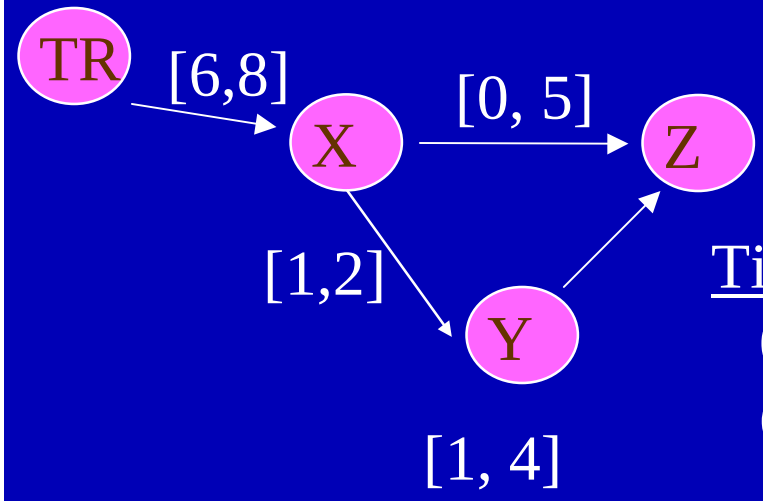
- The all-pairs, shortest path graph (the d-graph) for any STP is decomposable.

The Dispatch Problem

- Given a (set of) plan(s) with temporal constraints, decide when to execute each action.
- *For now, assume the plans are encoded as STPs.*

Naïve Dispatch Algorithm for STPs

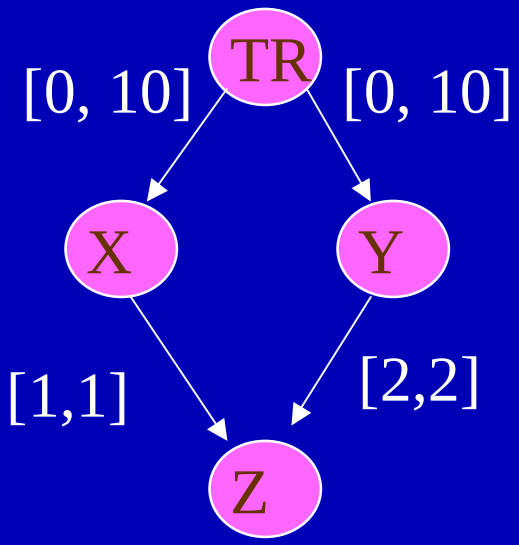
1. Select an event such that the current time is in its time window, and it's enabled.
2. Assign the current time to the selected event.
3. Propagate that assignment through the STP



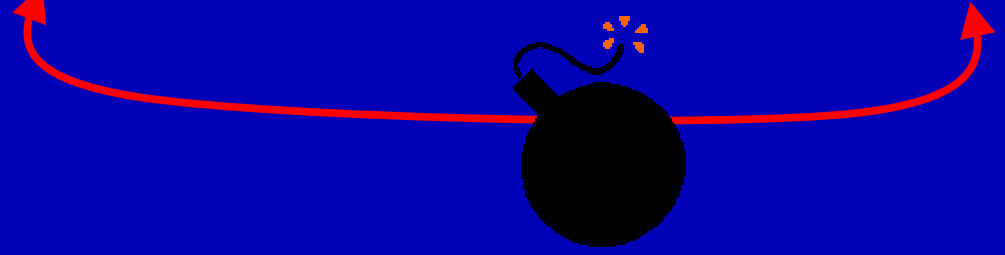
Time	Assign	TWs
0		X:<6,8> Y:<7,10> Z:<8,13>
6	X ← 6	X:<6,6> Y:<7,8> Z: <8,11>
7	Y ← 7	X:<6,6> Y:<7,7> Z: <8,11>
9	Z ← 7	X:<6,6> Y:<7,7> Z: <9,9>

Naïve Dispatch Algorithm for STPs

1. Select an event such that the current time is in its time window, and it's enabled.
2. Assign the current time to the selected event.
3. Propagate that assignment through the STP



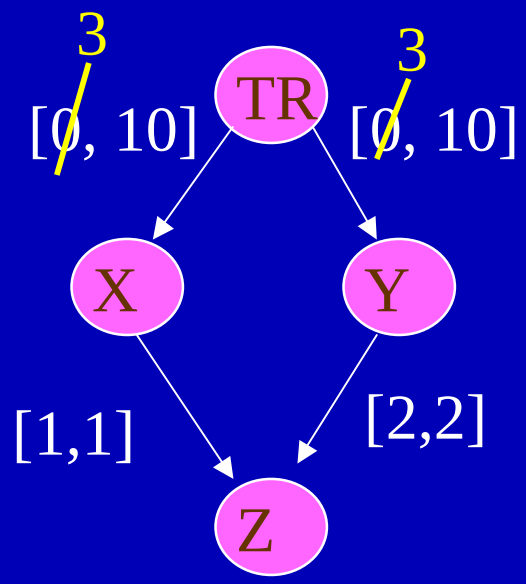
Time	Assign	TWs
0		X:<1,10> Y:<0,9>
3	X ← 3	X: <3,3> Y: <2,2>



Solution #1

Before making an assignment, set the lower bounds of all TWs to "*now*" and propagate.

Works, but requires O(e + n *log* n) work to propagate.

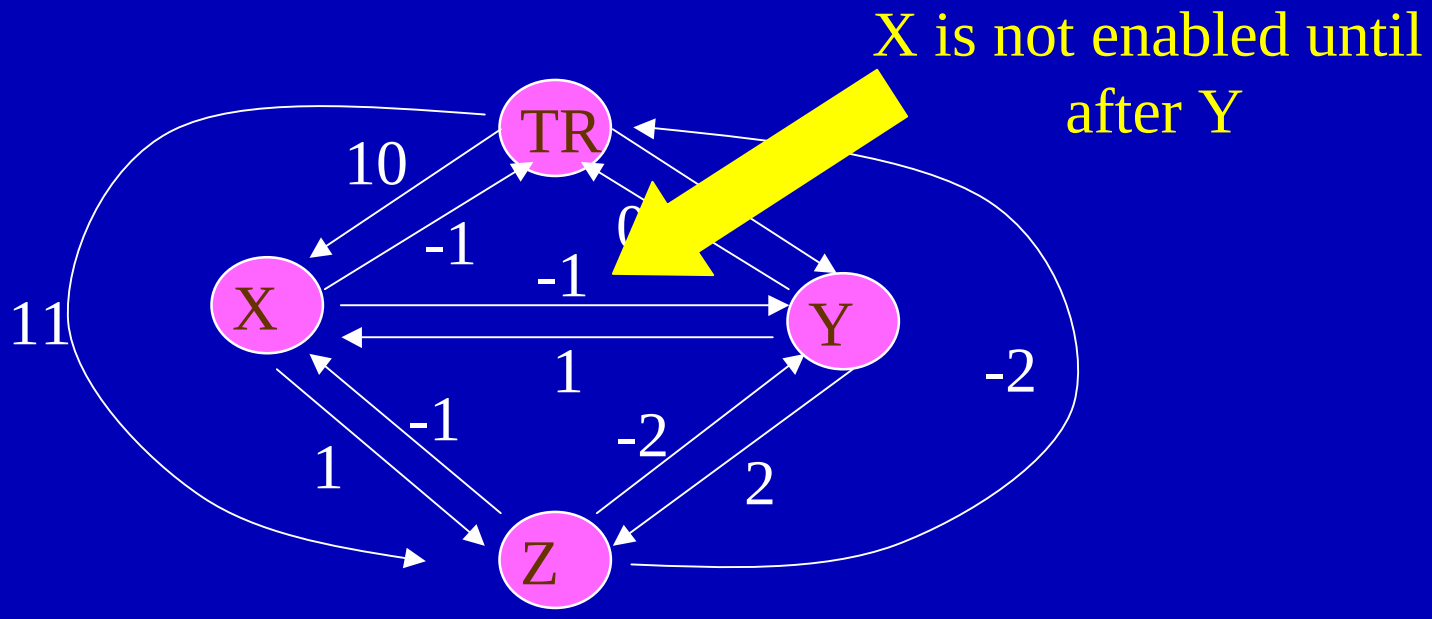


Time	Assign	TWs
0		X:<1,10> Y:<0,9>
3	{propagate}	X:<4,10> Y:<3,9>
3	Y ← 3	X: <4,4> Y:<3,3>

Solution #2

Compute the all-paths shortest pair network (off-line)—can then do only local propagation (to neighbors).

Works, and requires less propagation (exactly n), but can still do better.



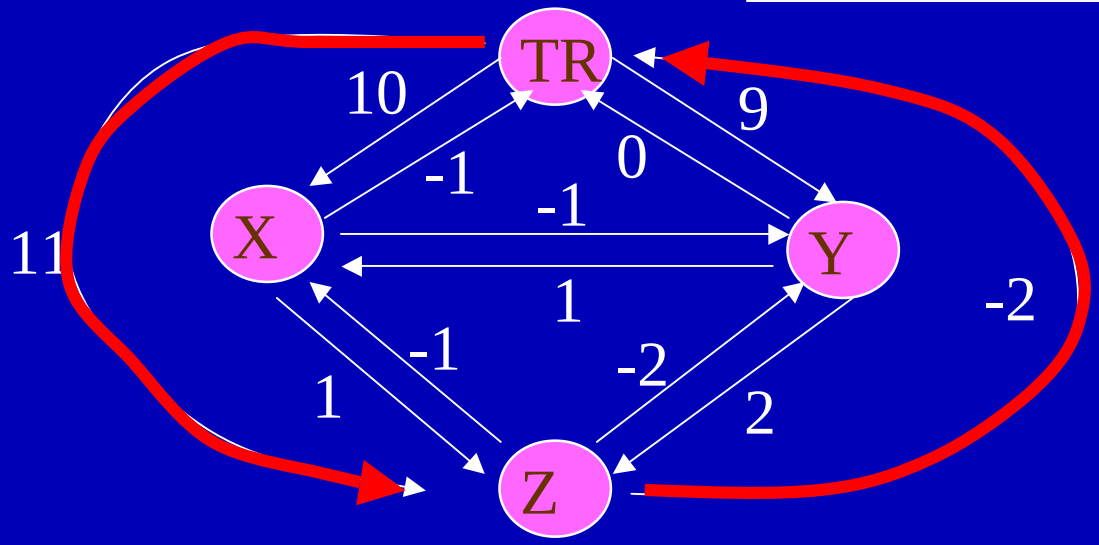
Solution #3

Compute the all-paths shortest pair network and then minimize it by deleting unneeded edges(off-line).

Triangle Rule: Edge AC is dominated if there is another node B such that:

- $|AB| + |BC| = |AC| \wedge$
- $\{ |AB| < 0 \quad \vee \quad |BC| \geq 0 \}$

NASA Remote Agent Domain
Pruned 40-70% of Nodes



Dispatch Algorithm

After computing an equivalent, minimal dispatchable STP, set A to the initially enabled events and:

1. Pick and remove an event *e* from A such that *now* $\in$ TW(E)
2. S $\leftarrow$ S $\cup$ {*e*}
3. Execution-time(*e*) $\leftarrow$ *now*
4. Propagate this assignment to the immediate neighbors of *e*
5. A $\leftarrow$ A $\cup$ {*x* | all negative edges starting at *x* have destinations already in S} ((I.e., *x* is enabled.))
6. Wait until *now* has advanced to some time between the minimum lower bound of a time window for a member of A and the minimum upper bound of a time window for a member of A.
7. Loop to (2) until every event is in S.

Today's Outline

√ 1. Introduction

2. Satisfying Temporal Constraints during Execution

   √ A. Temporal Constraint Networks: Simple Temporal Networks

   √ B. Execution Management (Plan Dispatch) for STPs

   C. More Expressive Networks: Disjunctive Temporal Networks

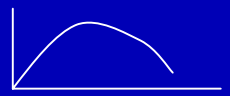
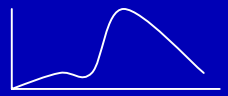
   D. Plan Dispatch and Plan Update for DTPs

   E. Yet More Expressive Networks

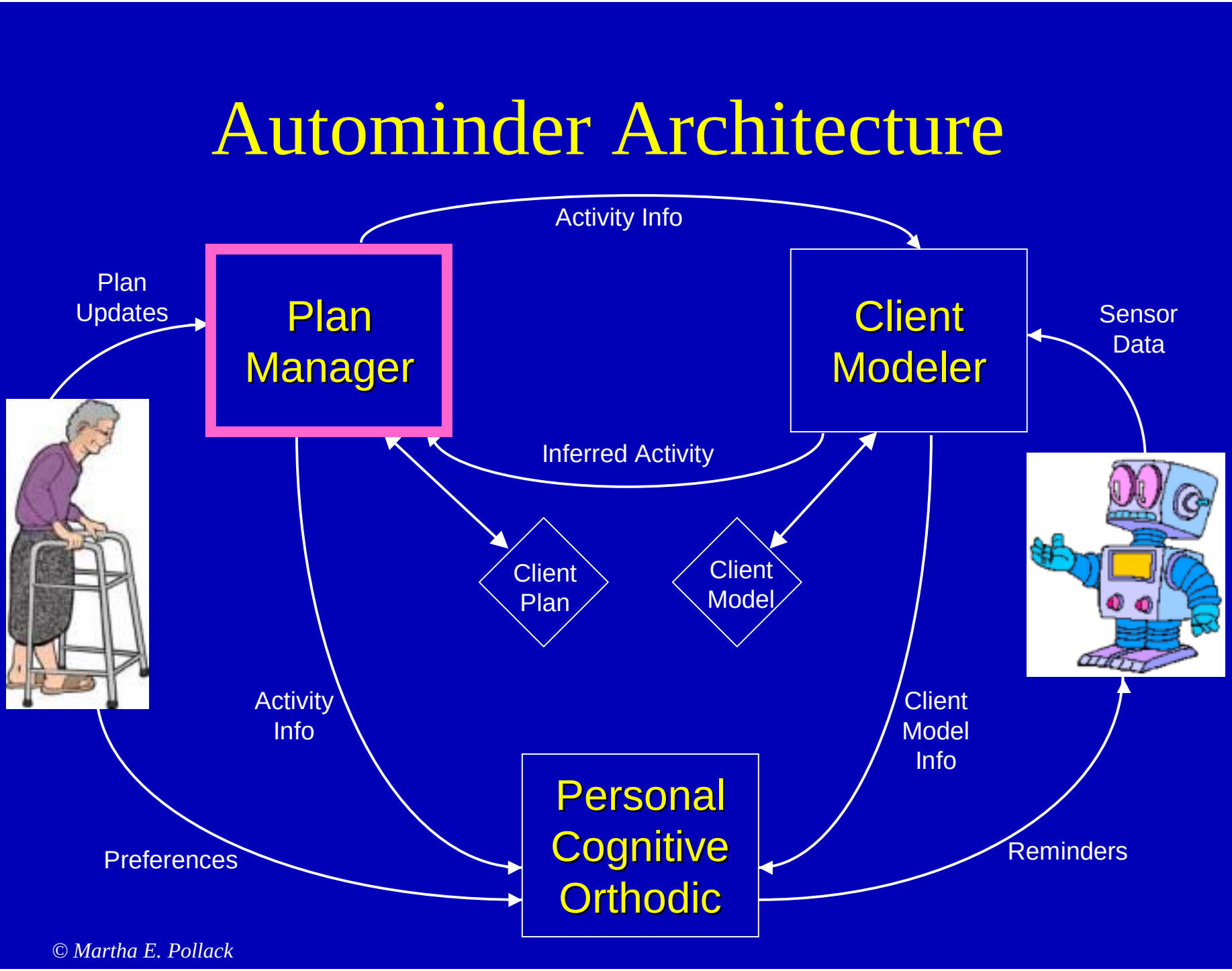
An Plan Management System

- The “Nursebot” Project (Initiative on Personal Robotics for the Elderly)
- Designing a *cognitive orthotic* (Autominder) to assist older adults with memory decline
- Currently deployed on a mobile robot (Pearl)

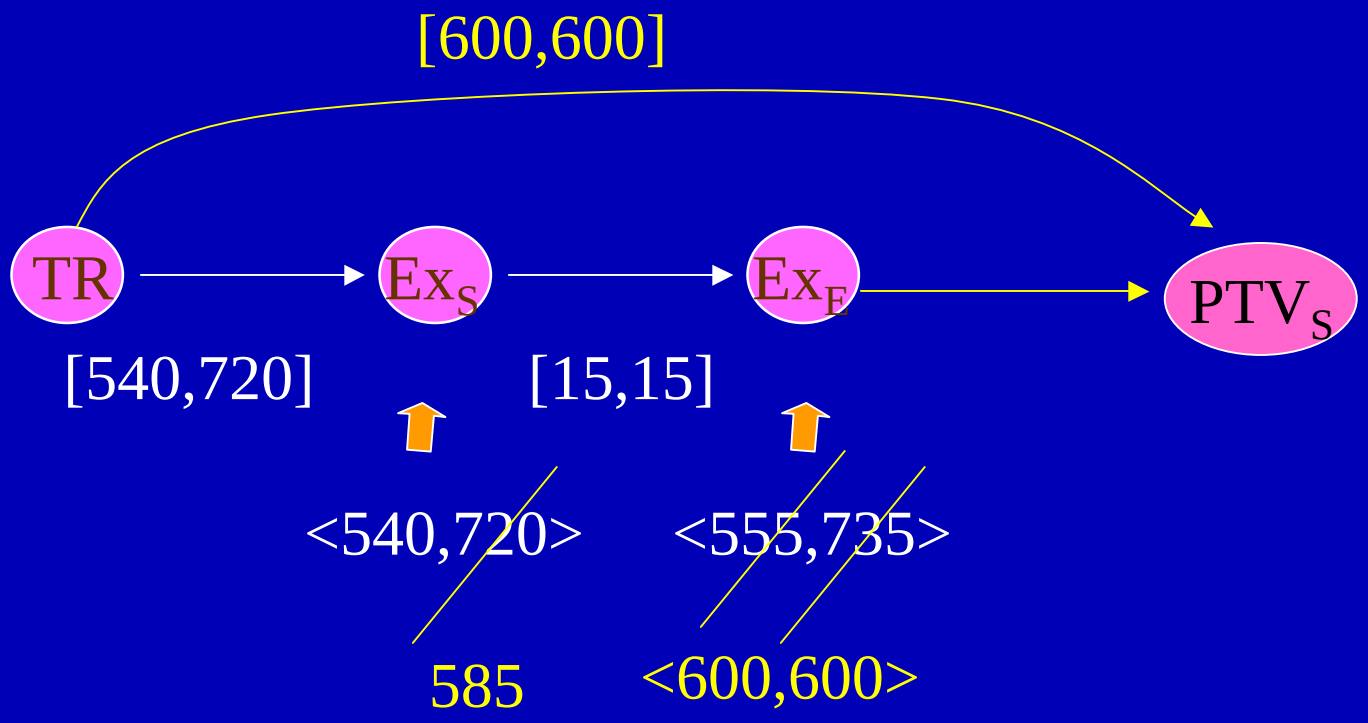
Autominder Example

Req/Opt	Activity	Allowed	Expected	Observed
R	toilet use	10:45-11:05		10:55
R	lunch	12:00-12:45		**REMIND** 12:28 **12:29**
O	TV	14:00-14:30		
R	toilet use	13:55-14:15		**REMIND 13:55**

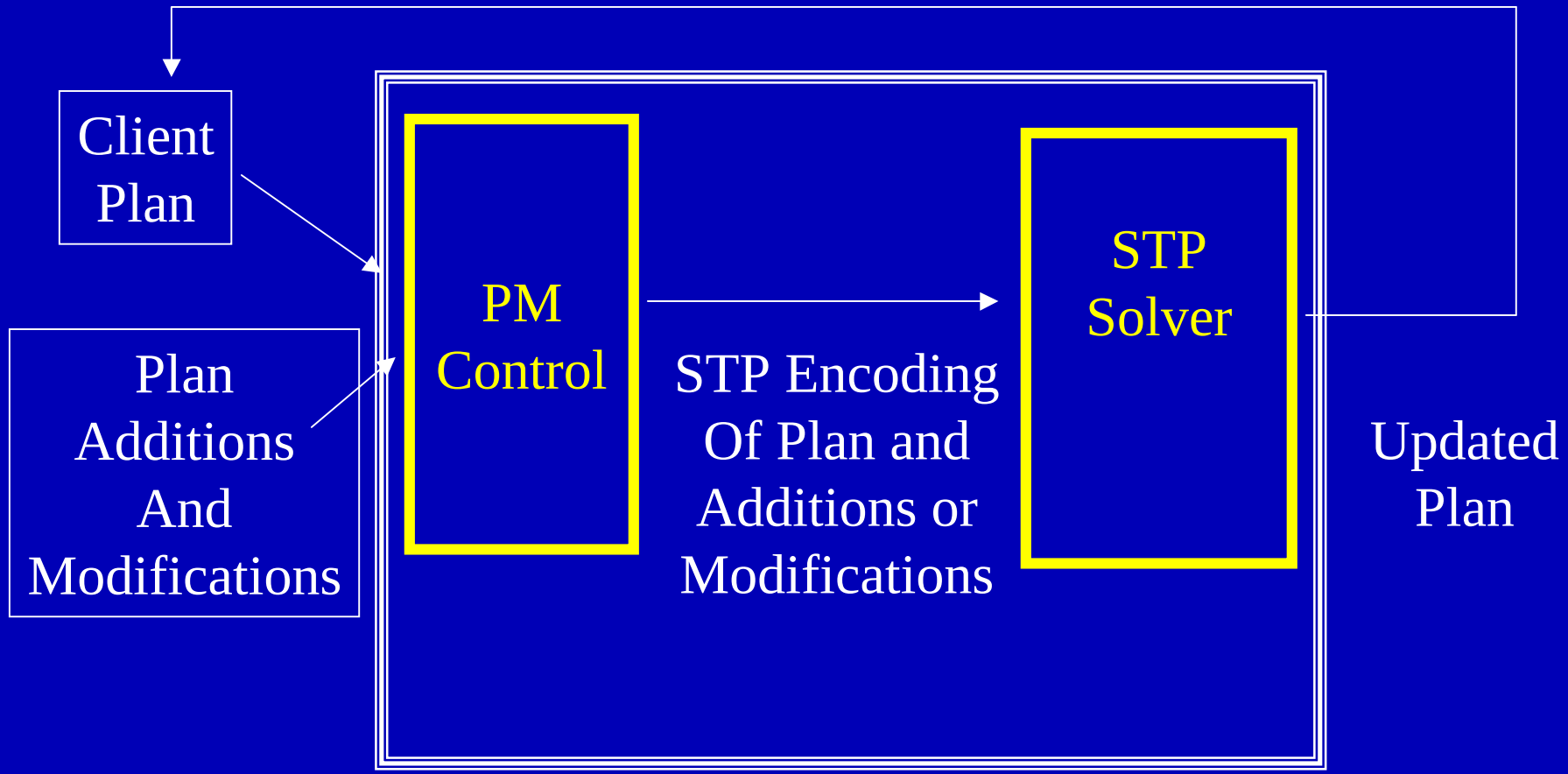
Autominder Architecture



Dispatch (and Update) Example I



Plan Management Architecture



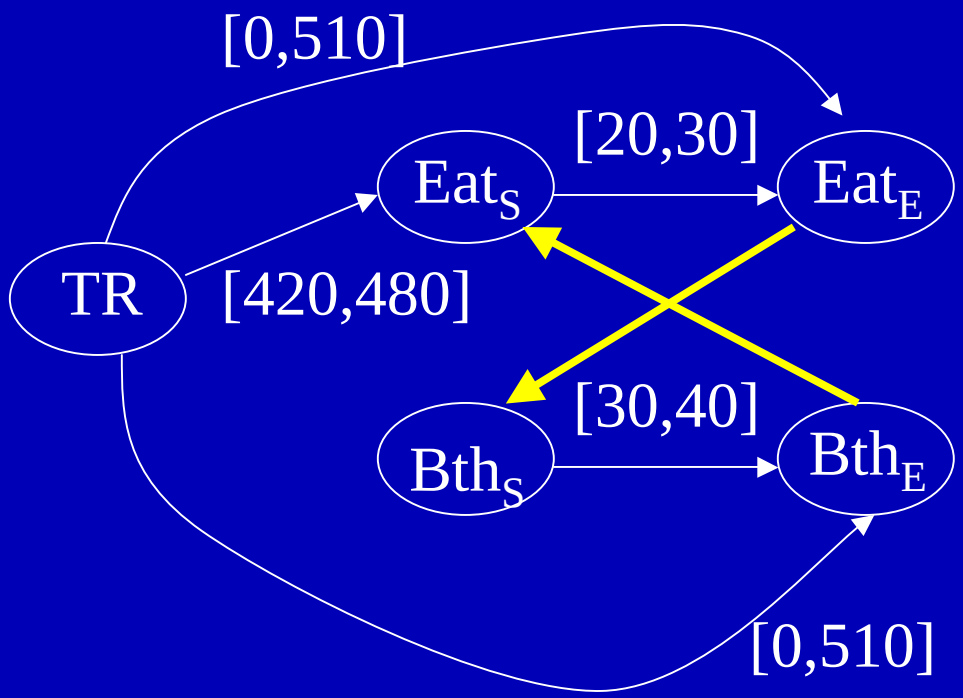
The Need for More Expressiveness

Client should start breakfast between 7 and 8 a.m., and breakfast will take 20 to 30 minutes.

Client should bathe, which will take 30 to 40 minutes.

Both breakfast and bathing must end by 8:30 a.m.

These tasks cannot overlap.



Disjunctive constraint:

$0 \leq Eat_S - Bth_E$ OR

$0 \leq Bth_S - Eat_E$

Disjunctive Temporal Problems

- A set of time points (variables) V and a set of constraints C of the form:

$$lb_{ji} \leq X_i - X_j \leq ub_{ji} \vee \ldots \vee lb_{mk} \leq X_k - X_m \leq ub_{mk}$$

- Solution: assignment of times to all variables, so that all constraints in C are satisfied (consistent).
- Generalization of the STPs and TCSPs

DTPs as CSPs

- One-Level Approach
  - Attempt direct assignment of times to DTP variables.
  - Limitations: inefficient; produces overconstrained solution
- Two-Level Approach
  - CSP variables: DTP constraints.
  - CSP values: Disjuncts from DTP constraints.
  - The constraints among the variables are implied by the semantics of the values, (not explicitly given).

DTP Solving Example

$C_1 : \{c_{11} : y - x \leq 5\}$

$C_2 : \{c_{21} : w - y \leq 5\} \vee \{c_{22} : x - y \leq -10\} \vee \{c_{23} : z - y \leq 5\}$

$C_3 : \{c_{31} : y - w \leq -10\}$

NO: $c_{21} \wedge c_{31}$

NO: $c_{11} \wedge c_{22}$

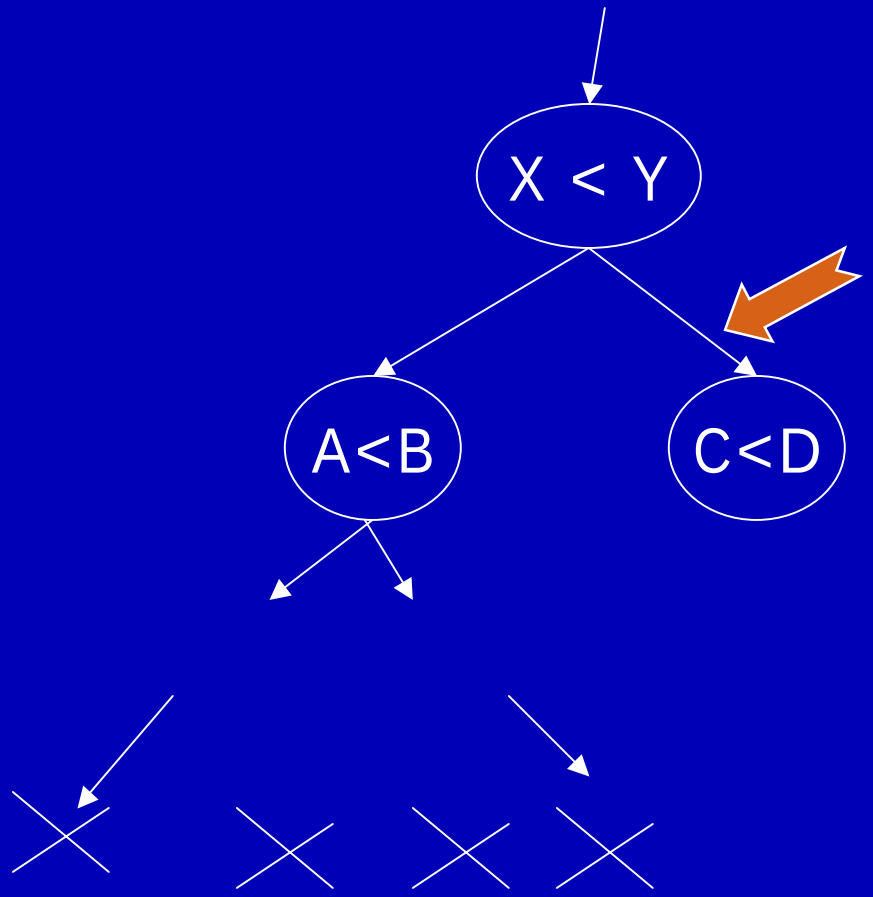
Component STP: $C_1 \leftarrow c_{11}, C_2 \leftarrow c_{23}, C_3 \leftarrow c_{31}$

One exact solution: $\{x = 0, y = 1, z = 2, w = 12\}$

Strategies for Efficiency

- Make the binary inconsistencies explicit, then use arc-consistency
- Use forward checking
- Use semantic branching
- Use conflict-directed backjumping
- Use incremental forward checking
- For the first layer use SAT solvers instead of CSP solvers
- Remove subsumed variables
- Introduce no-good learning

Semantic Branching



Also impose B≤A
(I.e. ¬ A<B)

Dispatch (and Update) Example II

OLD PLAN:

- Client should start breakfast between 7 and 8 a.m., and breakfast will take 20 to 30 minutes.
- Client should bathe, which will take 30 to 40 minutes.
- Both breakfast and bathing must end by 8:30 a.m
- These tasks cannot overlap.

Dispatch (and Update) Example II

OLD PLAN:

- Client should start breakfast between 7 and 8 a.m., and breakfast will take 20 to 30 minutes.
- Client should bathe, which will take 30 to 40 minutes.
- Both breakfast and bathing must end by 8:30 a.m
- These tasks cannot overlap.

ADD:

- Client should call her son at 7:30, for a 5 minute check-in.
- This cannot overlap with breakfast or bathing.

Dispatch (and Update) Example II

OLD PLAN:

•Client should start breakfast between 7 and 8 a.m., and breakfast will take 20 to 30 minutes.

•Client should bathe, which will take 30 to 40 minutes.

•Both breakfast and bathing must end by 8:30 a.m

•These tasks cannot overlap.

ADD:

•Client should call her son at 7:30, for a 5 minute check-in.

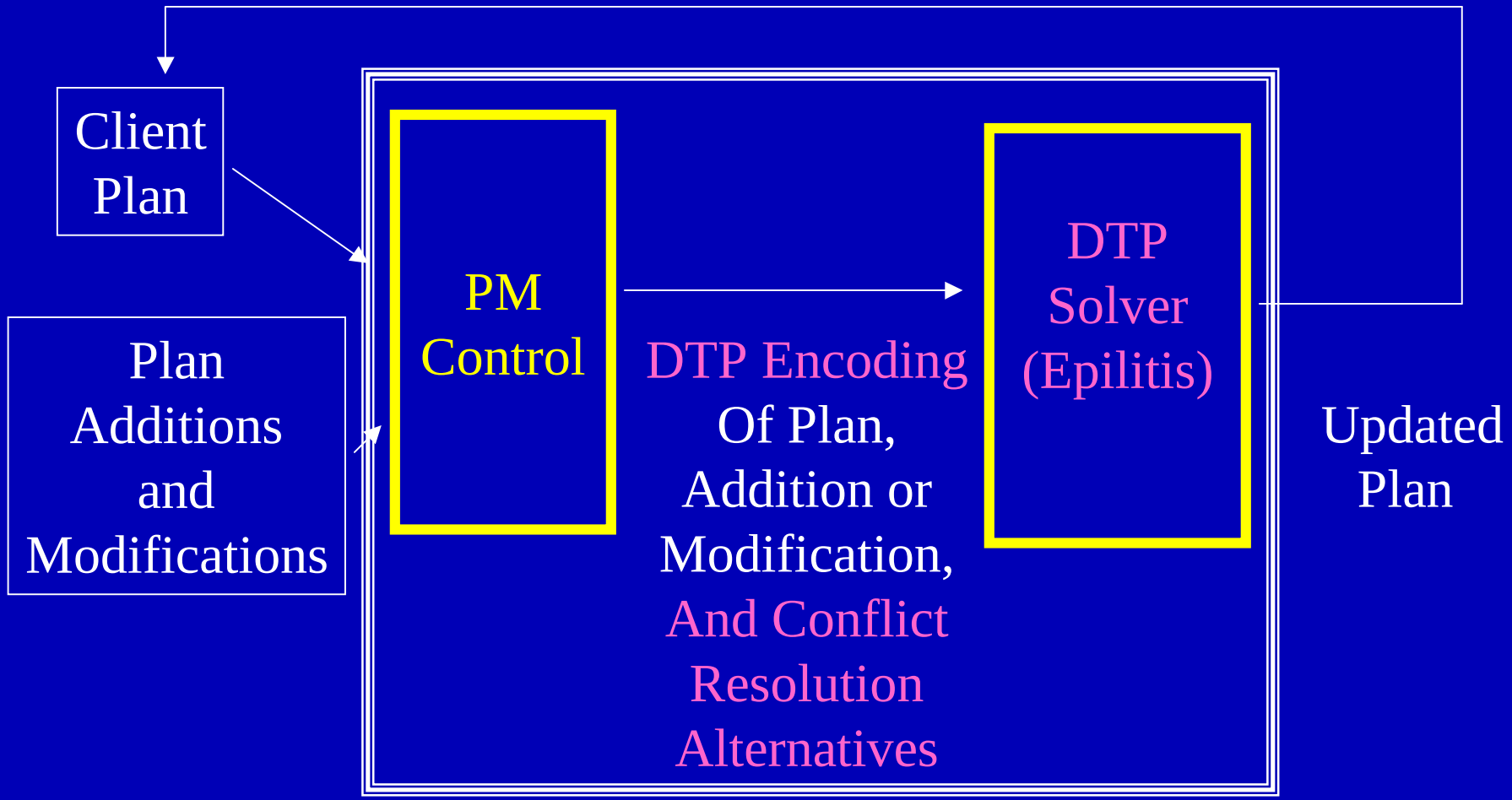
•This cannot overlap with breakfast or bathing.

DERIVE:

•Start breakfast by 7:10 or after 7:35; start bath by 7:00 or after 7:35; start one or the other by 7:40, adjusting maximum durations.

Plan Manager Architecture

[New Activity or Plan Modification]



Dispatch Example III

OLD PLAN:

- Client should start breakfast between 7 and 8 a.m., and breakfast will take 20 to 30 minutes.
- Client should bathe, which will take 30 to 40 minutes.
- Both breakfast and bathing must end by 8:30 a.m
- These tasks cannot overlap.

TIME BOUND PASSED:

- 7:30 passes with neither action begun.

Dispatch Example III

OLD PLAN:

- Client should start breakfast between 7 and 8 a.m., and breakfast will take 20 to 30 minutes.
- Client should bathe, which will take 30 to 40 minutes.
- Both breakfast and bathing must end by 8:30 a.m
- These tasks cannot overlap.

TIME BOUND PASSED:

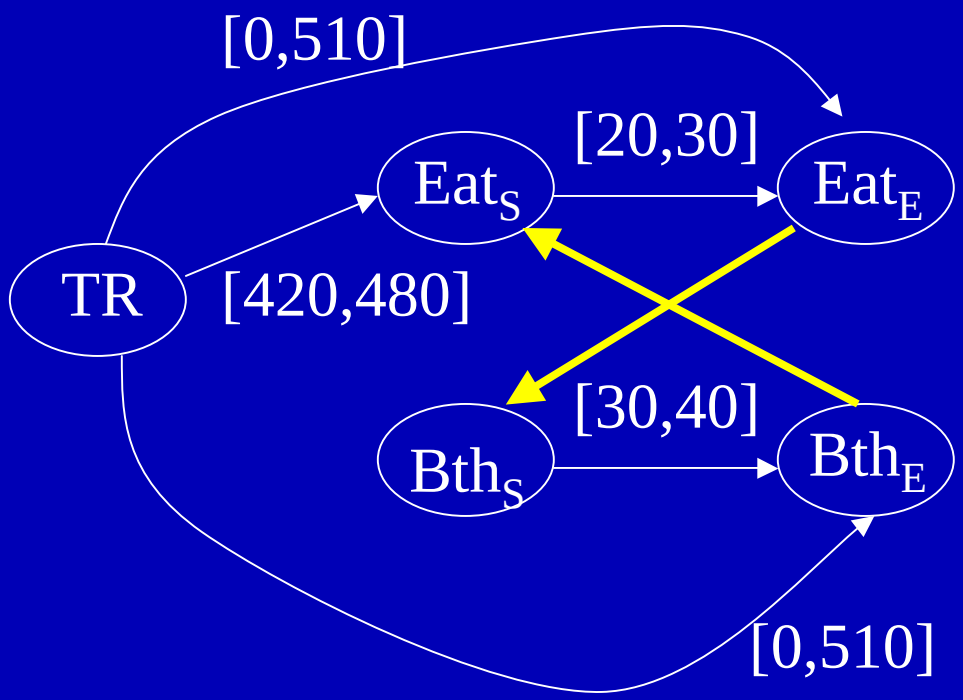
- 7:30 passes with neither action begun.

DERIVE:

- Don’t begin bath until after breakfast complete.

Execution Tracking Example

The one DTP . . .



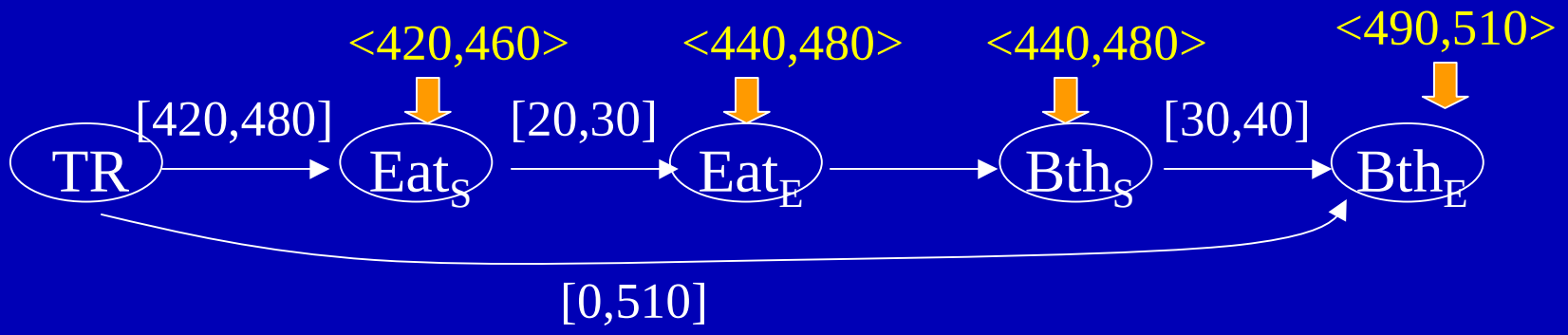
Disjunctive constraint:

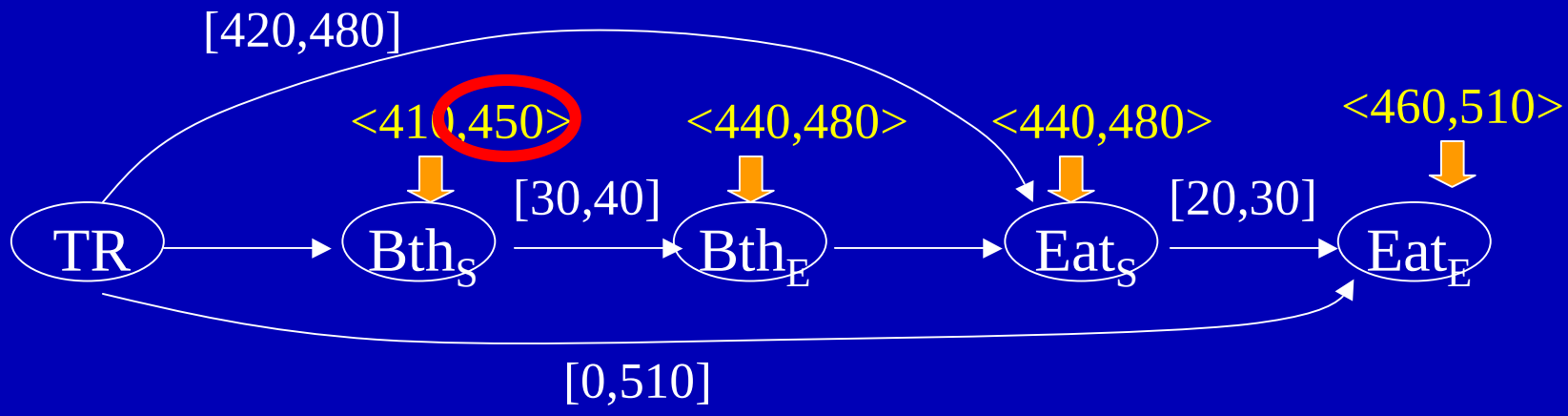
$0 \leq Eat_S - Bth_E$ OR

$0 \leq Bth_S - Eat_E$

Execution Tracking Example

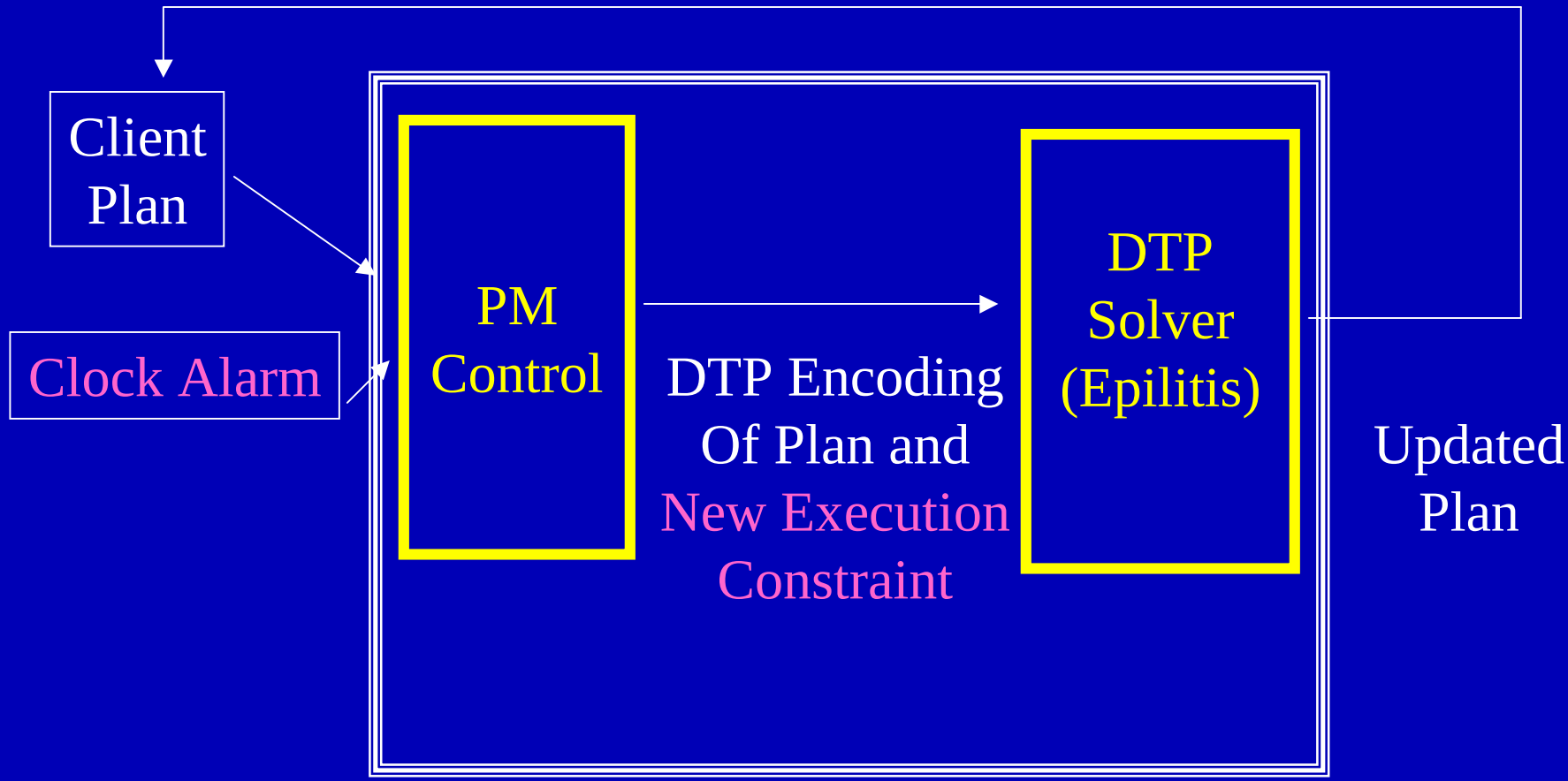
. . . Encodes 2 STPs





Plan Manager Architecture

[Time Bound Passed or Event Occurred]



Today's Outline

√ 1. Introduction

2. Satisfying Temporal Constraints during Execution

   √ A. Temporal Constraint Networks: Simple Temporal Networks

   √ B. Execution Management (Plan Dispatch) for STPs

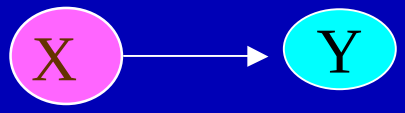
   √ C. More Expressive Networks: Disjunctive Temporal Networks

   √ D. Plan Dispatch and Plan Update for DTPs

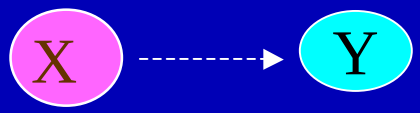
   E. Yet More Expressive Networks

Handling Temporal Uncertainty

- TP-u (e.g., STP-u)
- Distinguish between two kinds of events:
  - Controllable: the executing agent controls the time of occurrence
  - Uncontrollable: "nature" controls the time of occurrence



Controllable edge (Y controllable event)

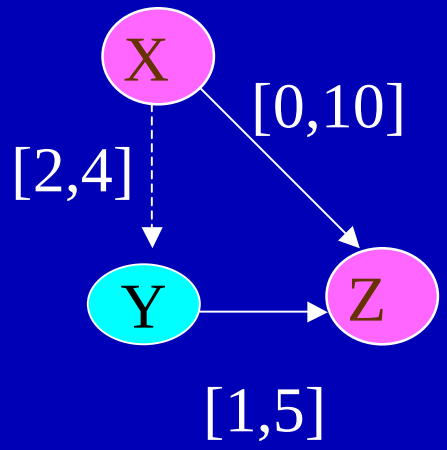


Uncontrollable edge (Y uncontrollable event)

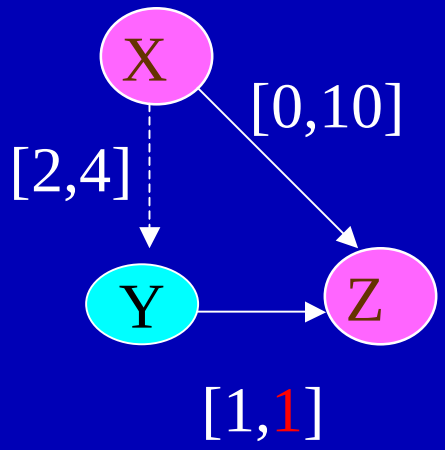
Three Notions of "Solution"

- *Strongly Controllable*: There is an assignment of time points to the controllable events such that the constraints will be satisfied regardless of when the uncontrollables occur.
- *Weakly Controllable*: For each outcome of the uncontrollables, there is an assignment of time points to the controllables such that the constraints are satisfied.
- *Dynamically Controllable*: As time progresses and uncontrollables occur, assignments can be made to the controllables such that the constraints are satisfied.

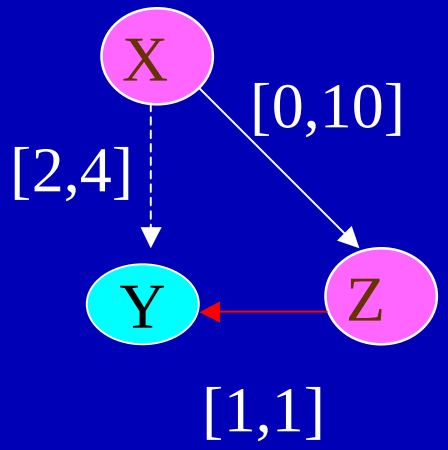
Controllability in STP-u's



Strongly Controllable
{X=0, Z = 5}



Dynamically Controllable
{X=0, Z = Y + 1}

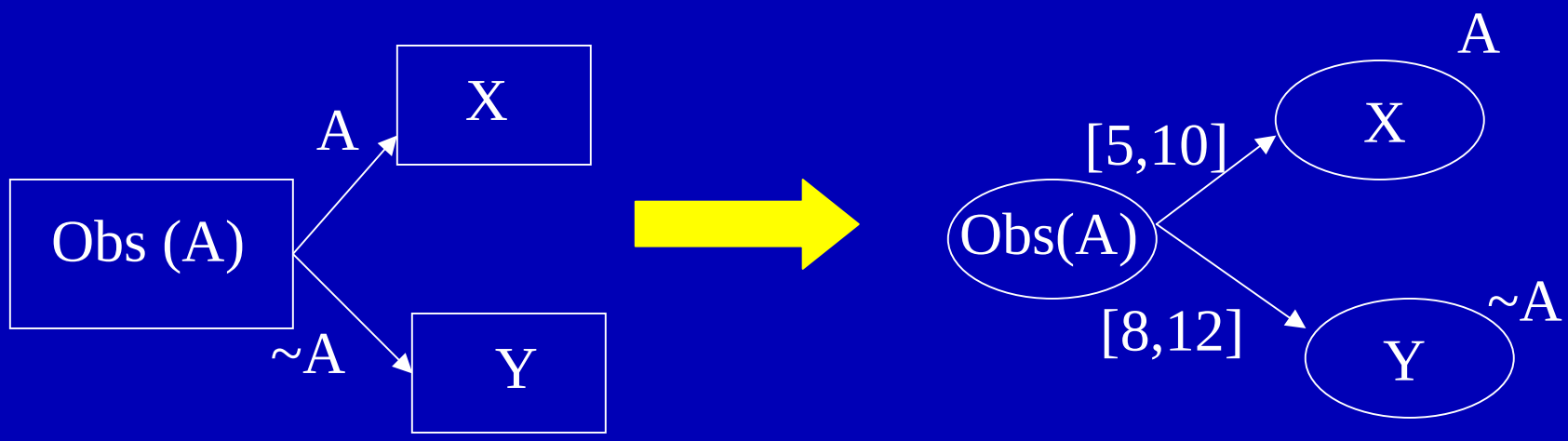


Weakly Controllable
{X=0, Z = Y – 1}

Strong => Dynamic => Weak

Handling Causal Uncertainty

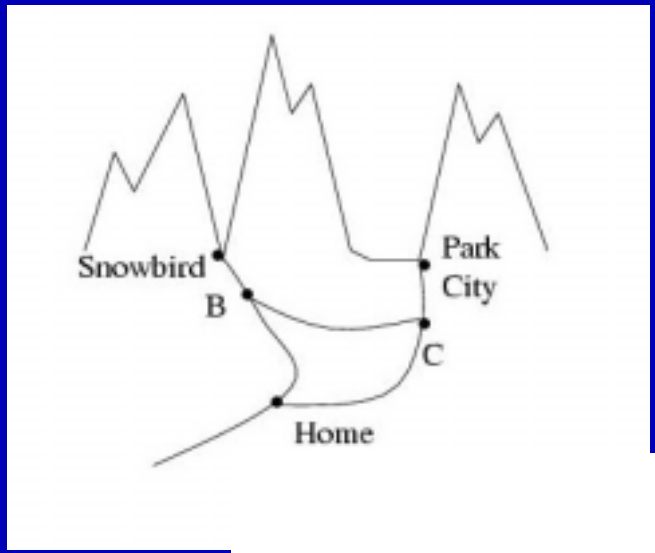
- CTP (e.g., CSTP)
- Label each node—events are executed only if their associated label is true (at a specified observation time)



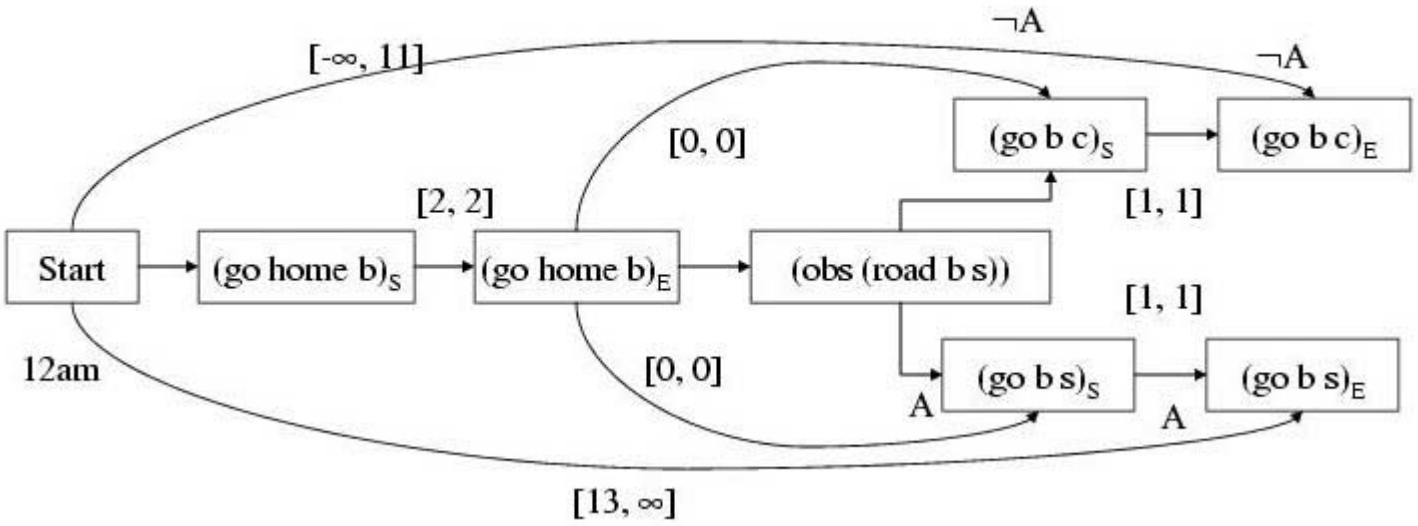
Conditional Plan

CTP

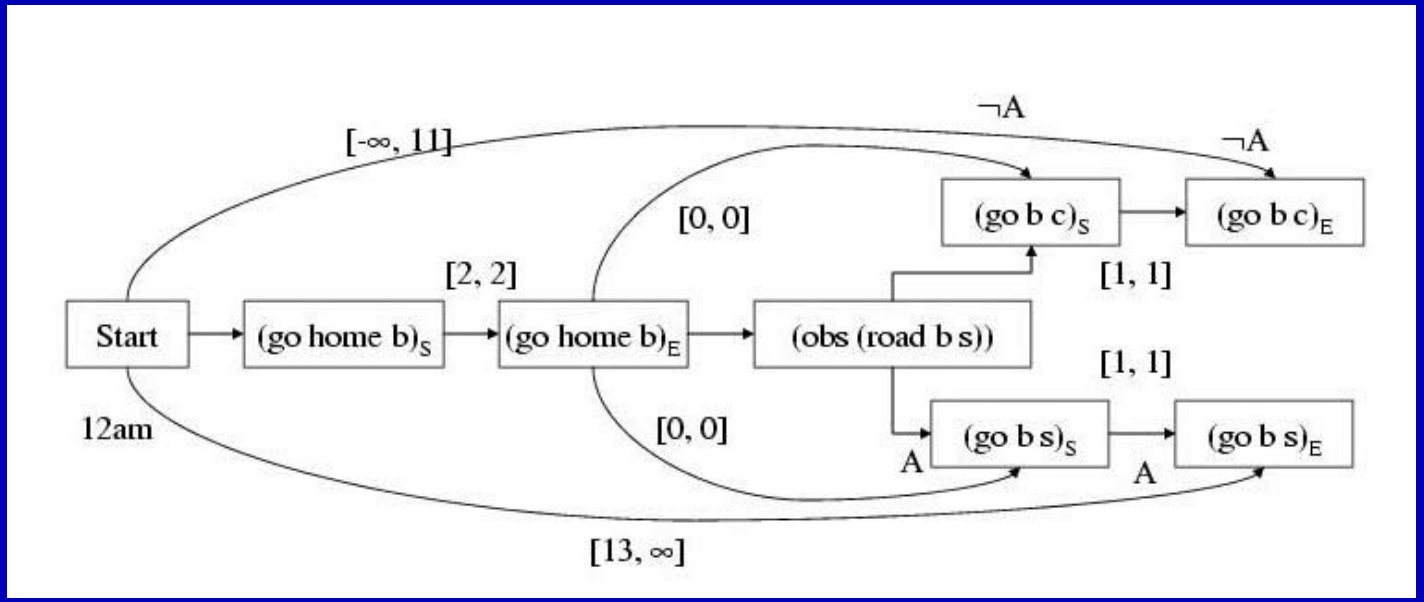
Conditional Plan as CTP



Travel from Home to S, but if the road is blocked from B to S, go to P.
If you go to S, arrive after 1p.m. (to take advantage of the discounts).
If you go to P, arrive at C by 11 a.m. (because traffic gets heavy).

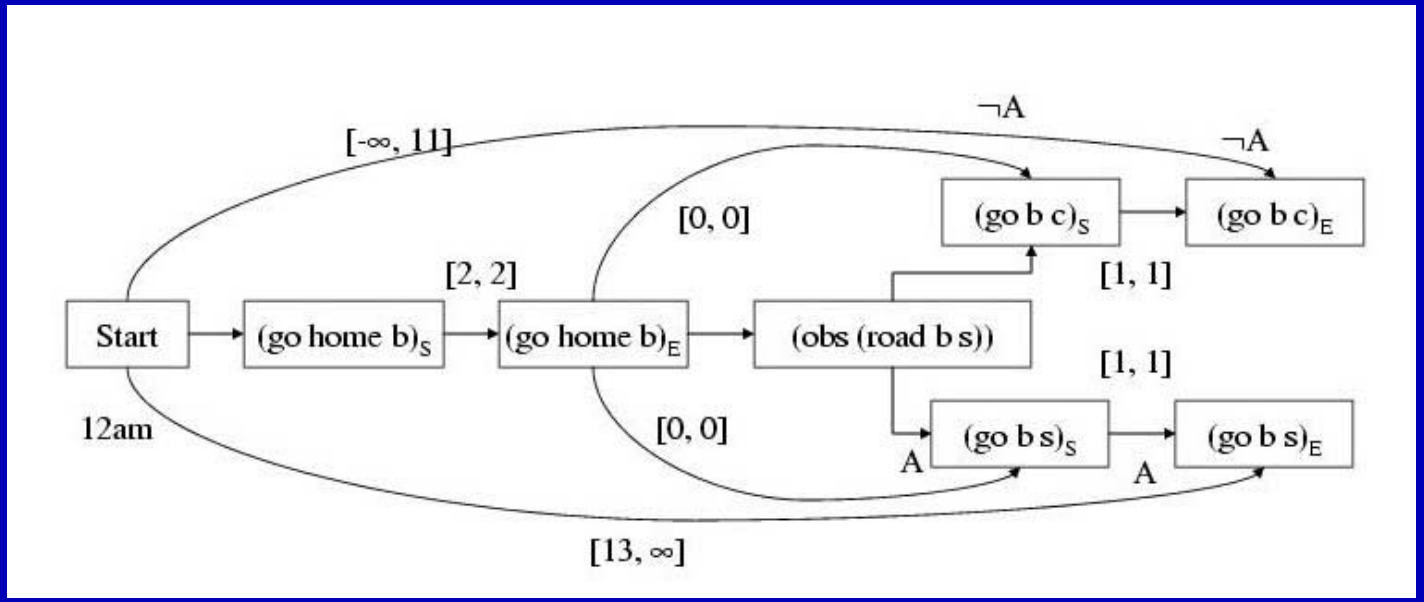


Strong Consistency



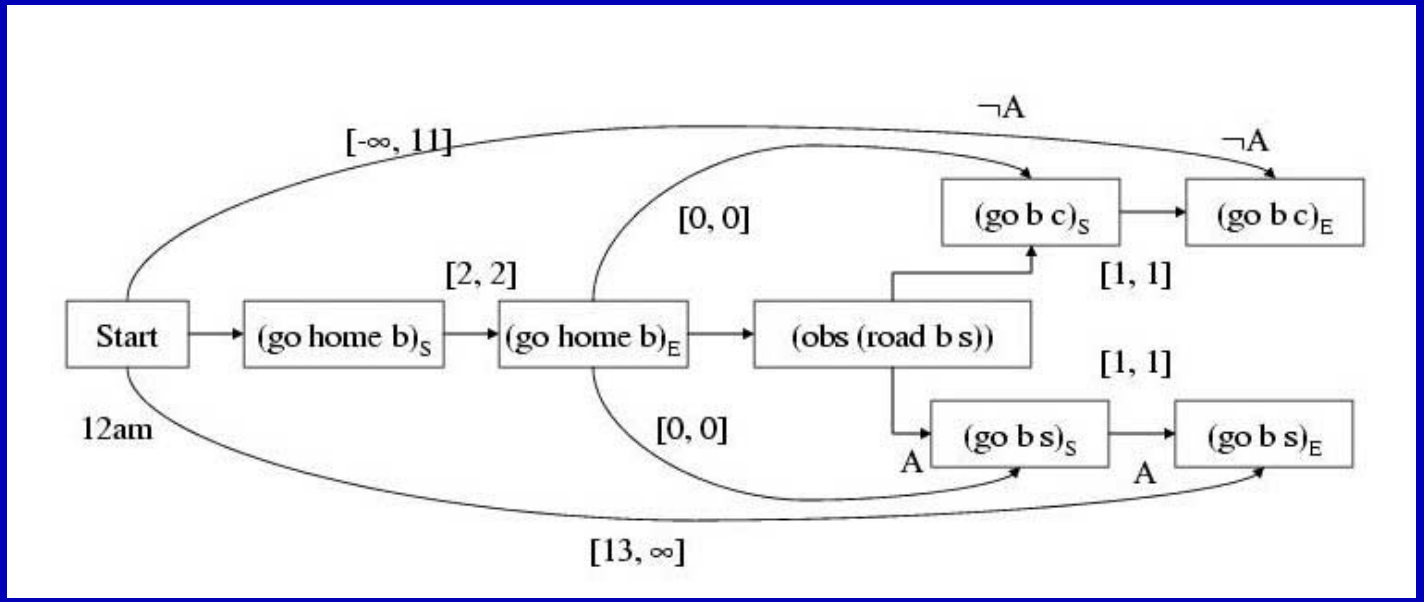
- Not strongly consistent: Must not be at B before 12 (if A is true); must be at B by 10 (if A is false)—and can't observe A until you're at B.

Weak Consistency



- Weakly consistent: When A is true, leave home after 10 (and all other assignments directly follow). When A is false, leave home by 9.

Dynamic Consistency



- Not dynamically consistent: Can't tell when you need to leave home until it's too late.
- Variant that is is dynamically consistent: Add a parking lot at B where you can wait.