Temporal Networks for Dynamic Scheduling

1st Summer School on Cognitive Robotics at MIT

Luke Hunsberger, Vassar College

June 12, 2017

Note

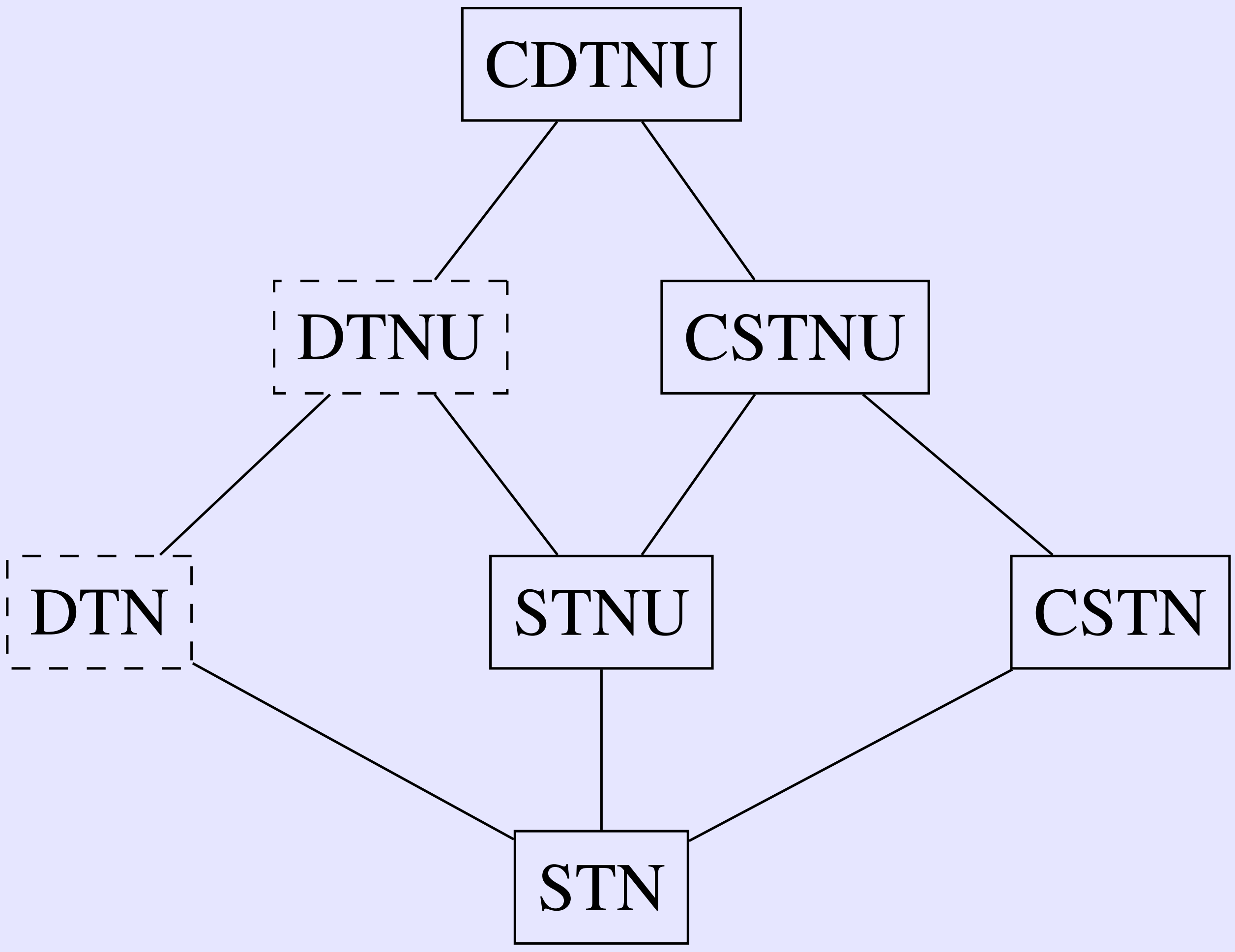
These slides have been edited from the version presented on June 12, 2017, as follows:

- Fixed the inequalities, $X - Z \leq 7$ and $Z - X \leq -3$, RE: “The *Zero* Time-Point”.
- Corrected the edge weights in the “Solving Sample STN” series.
- Inserted two slides “Remove Dominated Edges” that define “edge domination”.
- The edge from C to Z in “Making STN Dispatchable” is *not* dominated, and hence must remain in the “Dispatching the STN” series. Some edge lengths and time windows had to be corrected, too.

Outline

- Simple Temporal Networks (STNs)
- STNs with Uncertainty (STNUs)
- Conditional STNs (CSTNs)
- CSTNUs and beyond
- Conclusions

Temporal Networks



Simple Temporal Networks

Motivating Example

Goal: Fly from New York to Rome

- Leave New York after 4 p.m., June 8
- Return to New York before 10 p.m., June 18
- Away from New York no more than 7 days
- In Rome at least 5 days
- Return flight lasts no more than 7 hours

Simple Temporal Network (STN)*

- Includes time-points and temporal constraints
- Flexible: Time-points may "float"; not "nailed down" until they are *executed*
- Efficient algorithms for determining consistency, managing real-time execution, and handling new constraints

* (Dechter, Meiri, and Pearl 1991)

Simple Temporal Network*

A *Simple Temporal Network (STN)* is a pair, $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, where:

- $\mathcal{T}$ is a set of time-point variables: $\{t_1, \ldots, t_n\}$; and
- $\mathcal{C}$ is a set of binary constraints, each of the form:

  $t_j - t_i \leq \delta$, where δ is a real number.

* (Dechter, Meiri, and Pearl 1991)

The *Zero* Time-Point, Z

- It is useful to have one time-point, called Z, whose value is fixed at 0.
- Binary constraints involving Z are equivalent to unary constraints:

$$X - Z \leq 7 \quad \Longleftrightarrow \quad X \leq 7$$

$$Z - X \leq -3 \quad \Longleftrightarrow \quad X \geq 3$$

Basic Notions for STNs

- A *solution* to an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is a complete set of assignments to the time-points in $\mathcal{T}$:

$$\{t_1 = w_1,\ t_2 = w_2,\ \ldots,\ t_n = w_n\}$$

  that together satisfy all of the constraints in $\mathcal{C}$.

- An STN with at least one solution is *consistent.*

- STNs with identical solution sets are *equivalent*.

STN for Travel Example

$\mathcal{T} \; = \; \{Z, t_1, t_2, t_3, t_4\}, \qquad Z = \text{Noon, June 8.}$

$\mathcal{C} =$

$$\left\{ \begin{array}{rcrl} Z - t_1 & \leq & -4 & \text{(Lv NYC after 4 p.m., June 8)} \\ t_4 - Z & \leq & 250 & \text{(Av NYC by 10 p.m., June 18)} \\ t_4 - t_1 & \leq & 168 & \text{(Gone no more than 7 days)} \\ t_2 - t_3 & \leq & -120 & \text{(In Rome at least 5 days)} \\ t_4 - t_3 & \leq & 7 & \text{(Return flight less than 7 hrs)} \end{array} \right\}$$

Graph for an STN*

The *graph* for an STN, $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, is a graph, $\mathcal{G} = (\mathcal{T}, \mathcal{E})$, where:

- Time-points in $\mathcal{S}$ $\iff$ nodes in $\mathcal{G}$
- Constraints in $\mathcal{C}$ $\iff$ edges in $\mathcal{E}$:

$$Y - X \leq \delta \quad \iff \quad X \xrightarrow{\delta} Y$$

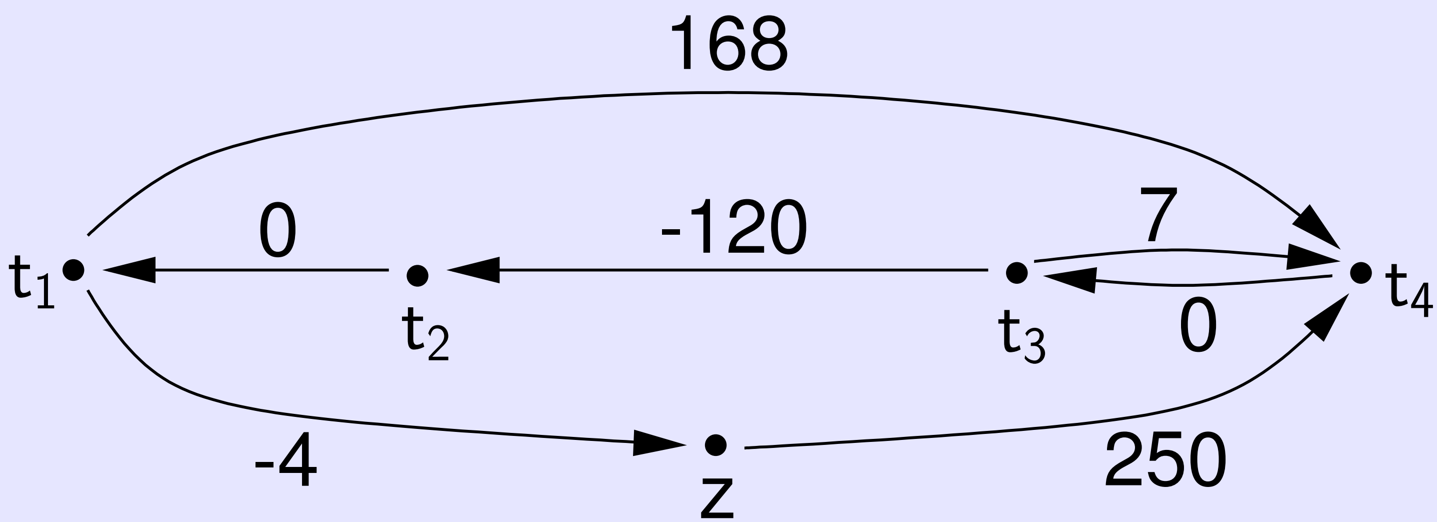
* (Dechter, Meiri, and Pearl 1991)

Graphical Representations

Constraint(s)	Edge(s)	Alt. Edge(s)
$3 \leq Y - X \leq 7$	$X \underset{-3}{\overset{7}{\rightleftarrows}} Y$	$X \xrightarrow{[3, 7]} Y$
$4 \leq X \leq 9$	$Z \underset{-4}{\overset{9}{\rightleftarrows}} X$	$Z \xrightarrow{[4, 9]} X$

Graph for Airline Scenario

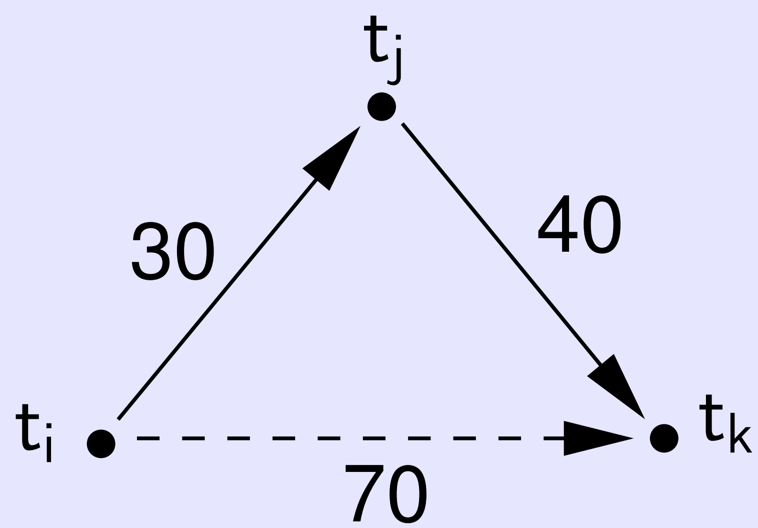
$$\left\{\begin{array}{rcrcrcr} Z - t_1 & \leq & -4, & \quad & t_4 - Z & \leq & 250 \\ t_4 - t_1 & \leq & 168, & & t_2 - t_3 & \leq & -120 \\ t_4 - t_3 & \leq & 7, & & t_1 - t_2 & \leq & 0 \\ t_3 - t_4 & \leq & 0 & & & & \end{array}\right\}$$



Implicit Constraints

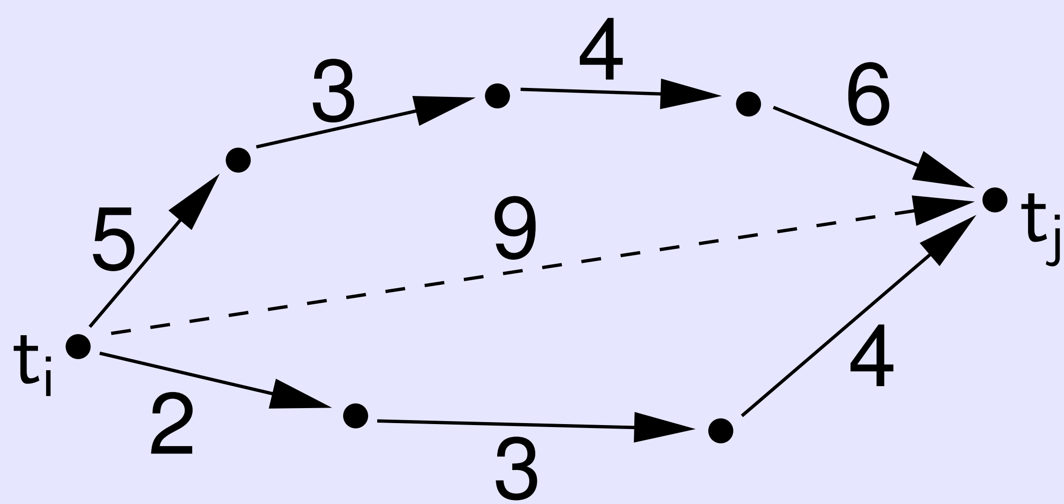
Explicit constraints combine to form implicit constraints:

$$\begin{array}{rcl} t_j - t_i & \leq & 30 \\ t_k - t_j & \leq & 40 \\ \hline t_k - t_i & \leq & 70 \end{array}$$



Chains of Constraints as Paths

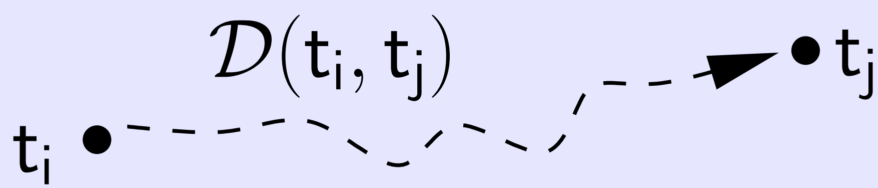
- Chains of constraints correspond to *paths* in the graph.
- *Stronger* constraints correspond to *shorter* paths.



Distance Matrix *

The *Distance Matrix* for an STN, $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, is a matrix $\mathcal{D}$ defined by:

$\mathcal{D}(t_i, t_j) \;=\;$ Length of Shortest Path from t_i to t_j in the graph for $\mathcal{S}$



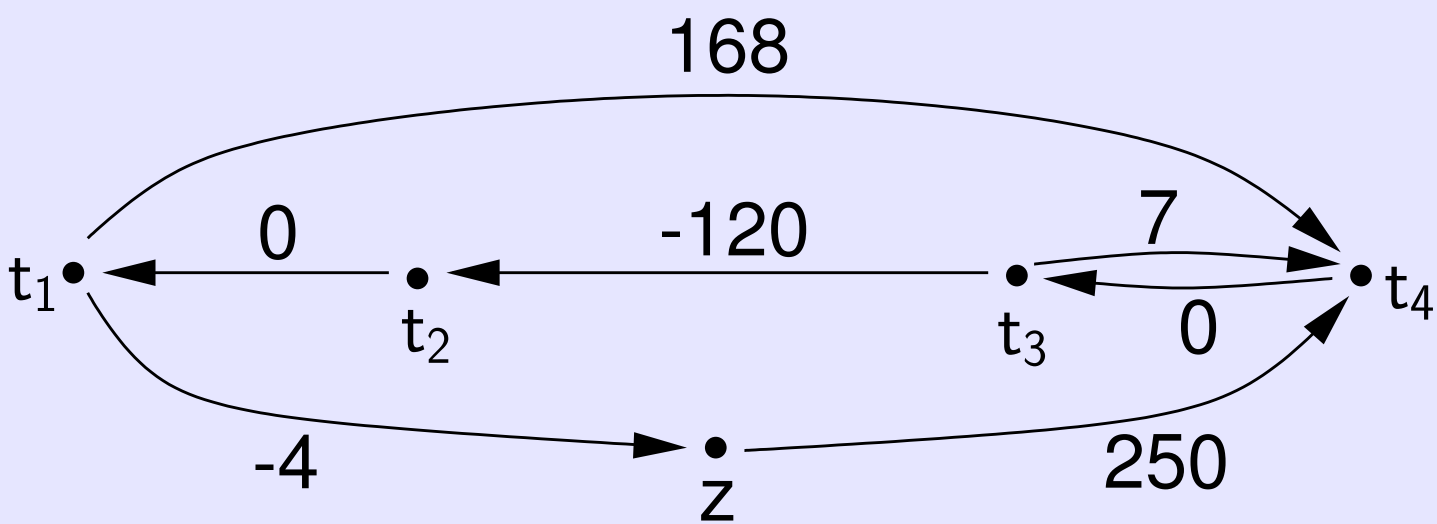
* (Dechter, Meiri, and Pearl 1991)

Distance Matrix (cont'd.)

- The strongest implicit constraint on t_i and t_j in $\mathcal{S}$ is:
  $$t_j - t_i \ \leq \ \mathcal{D}(t_i, t_j)$$
- $\mathcal{D}$ is the *All-Pairs, Shortest-Path* (APSP) Matrix for the STN's graph.*

* (Cormen, Leiserson, and Rivest 1990)

Travel Scenario's Distance Matrix



$\mathcal{D}$	z	t_1	t_2	t_3	t_4
z	0	130	130	250	250
t_1	-4	0	48	168	168
t_2	-4	0	0	168	168
t_3	-124	-120	-120	0	7
t_4	-124	-120	-120	0	0

Fundamental Theorem of STNs*

For an STN $\mathcal{S}$, with graph $\mathcal{G}$, and distance matrix $\mathcal{D}$, the following are equivalent:

- $\mathcal{S}$ is consistent
- $\mathcal{D}$ has non-negative values on main diagonal
- $\mathcal{G}$ has no negative-length loops

* (Dechter, Meiri, and Pearl 1991)

Computing $\mathcal{D}$ from Scratch

For an STN with n time-points and m edges:

- Floyd-Warshall Algorithm: $O(n^3)$
- Johnson's Algorithm: $O(n^2 \log n + nm)$

(Cormen, Leiserson, and Rivest 1990)

Dynamically Updating $\mathcal{D}$

- $O(n^2)$-time *incremental* algorithms update $\mathcal{D}$ in response to inserting a new constraint.

- $O(n^3)$-time *decremental* algorithms update $\mathcal{D}$ in response to weakening/deleting a constraint.

(Rohnert 1985; Even and Gazit 1985; Gerevini, Perini, and Ricci 1996; Ramalingam and Reps 1996; Cesta and Oddi 1996; Demetrescu and Italiano 2002)

Incremental Consistency

Verifying consistency after inserting/weakening constraints is less expensive than fully updating the distance matrix.*

- Algorithm maintains/updates a solution to the STN.
- Can verify consistency in $O(m + n \log n)$ time after inserting a new constraint.
- Deleting/weakening a constraint requires only constant time.

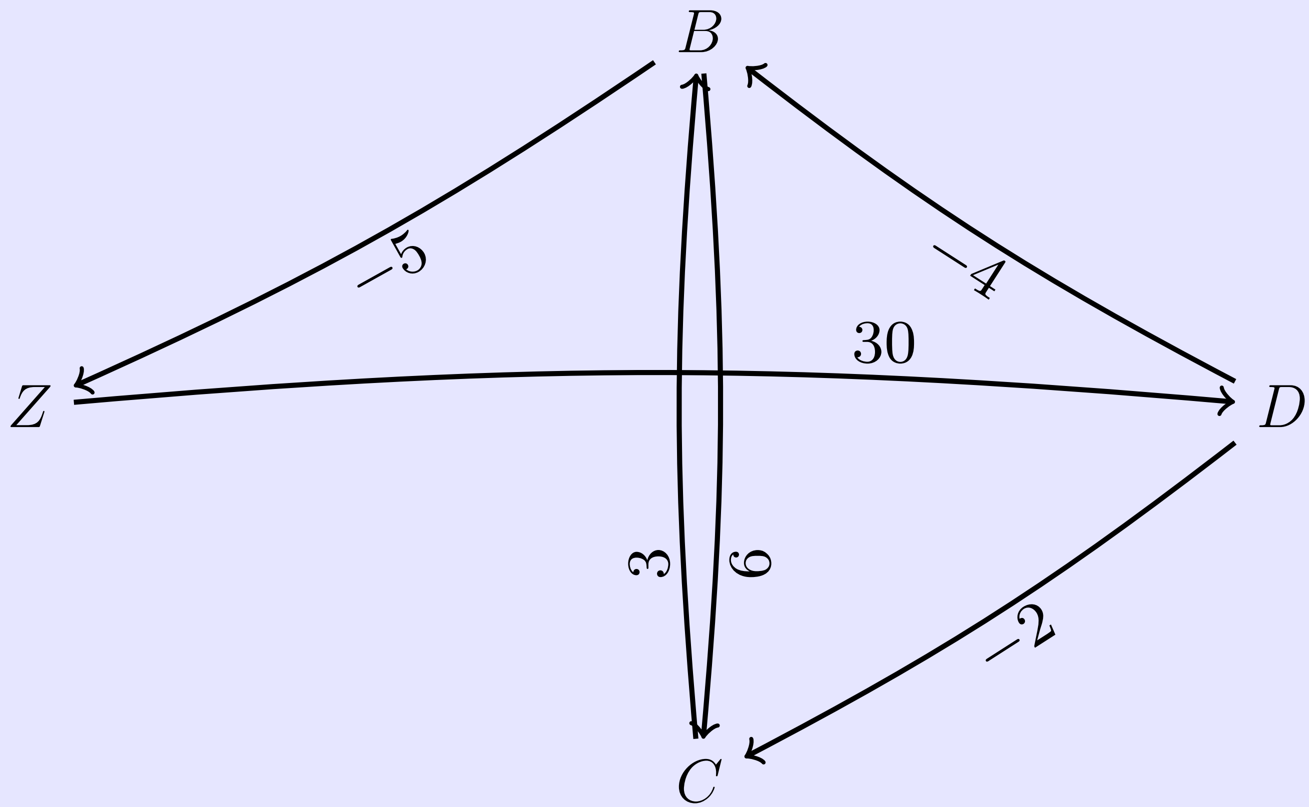
* (Ramalingam et al. 1999)

Finding a solution for an STN

- $\mathcal{D}$ has all necessary information.
- Time window for any X: $[-\mathcal{D}(X,Z),\ \mathcal{D}(Z,X)]$
- Simple algorithm to find a solution:
  - ○ Pick any time-point that doesn't yet have a value;
  - ○ Give it a value from its time-window;
  - ○ Update $\mathcal{D}$; $\Leftarrow$ expensive ...
  - ○ Repeat until all time-points have values.

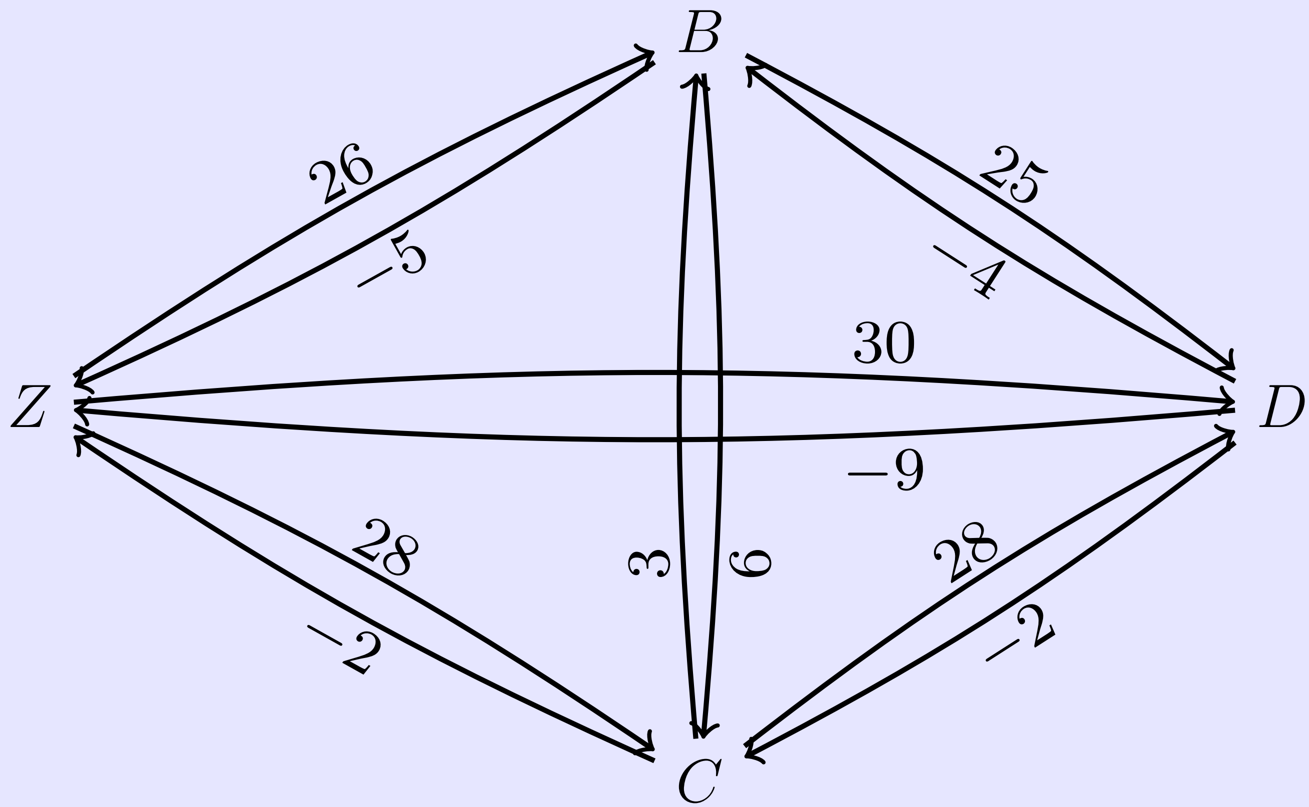
* (Dechter, Meiri, and Pearl 1991)

Sample STN



“Solving” Sample STN

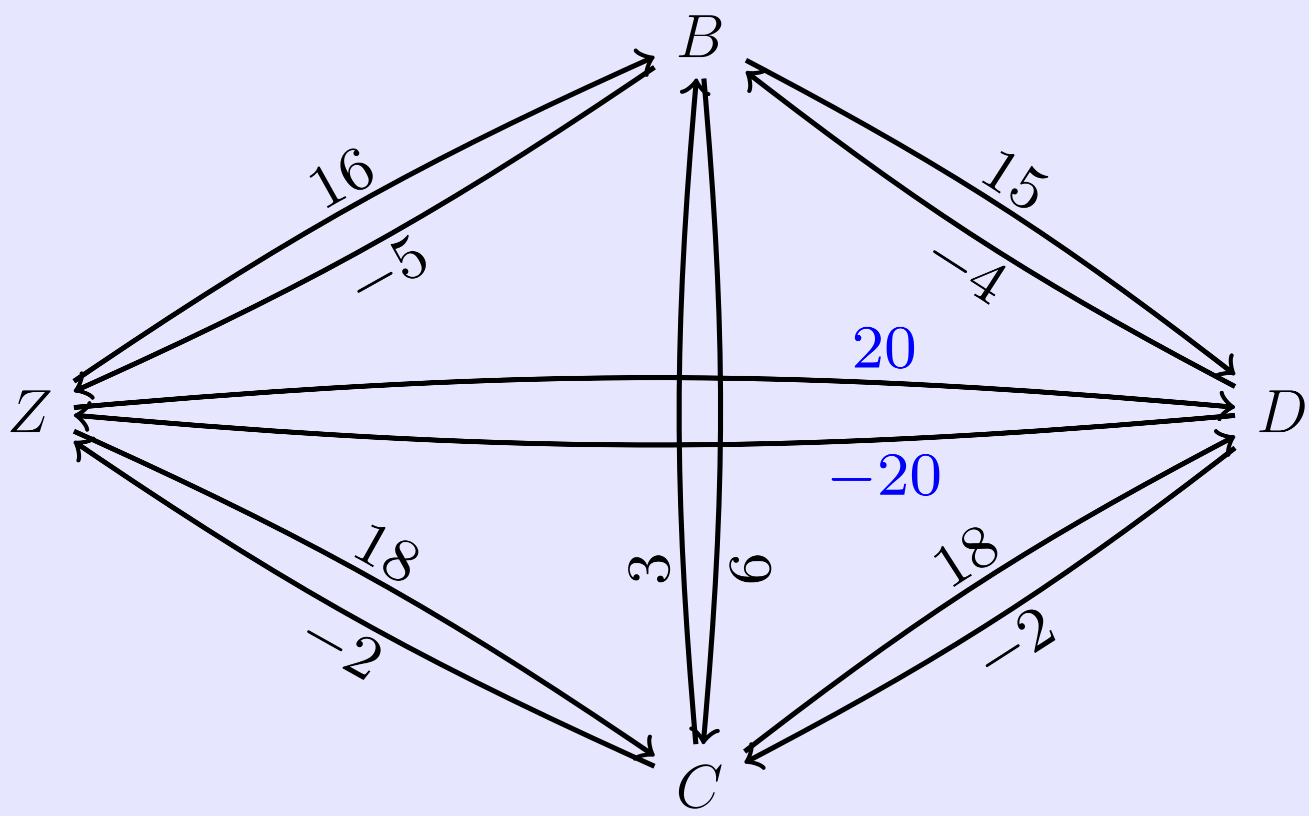
First, form APSP graph (equiv. compute $\mathcal{D}$).



Time Windows: $B \in [5, 26],\ C \in [2, 28],\ D \in [9, 30]$

“Solving” Sample STN (ctd.)

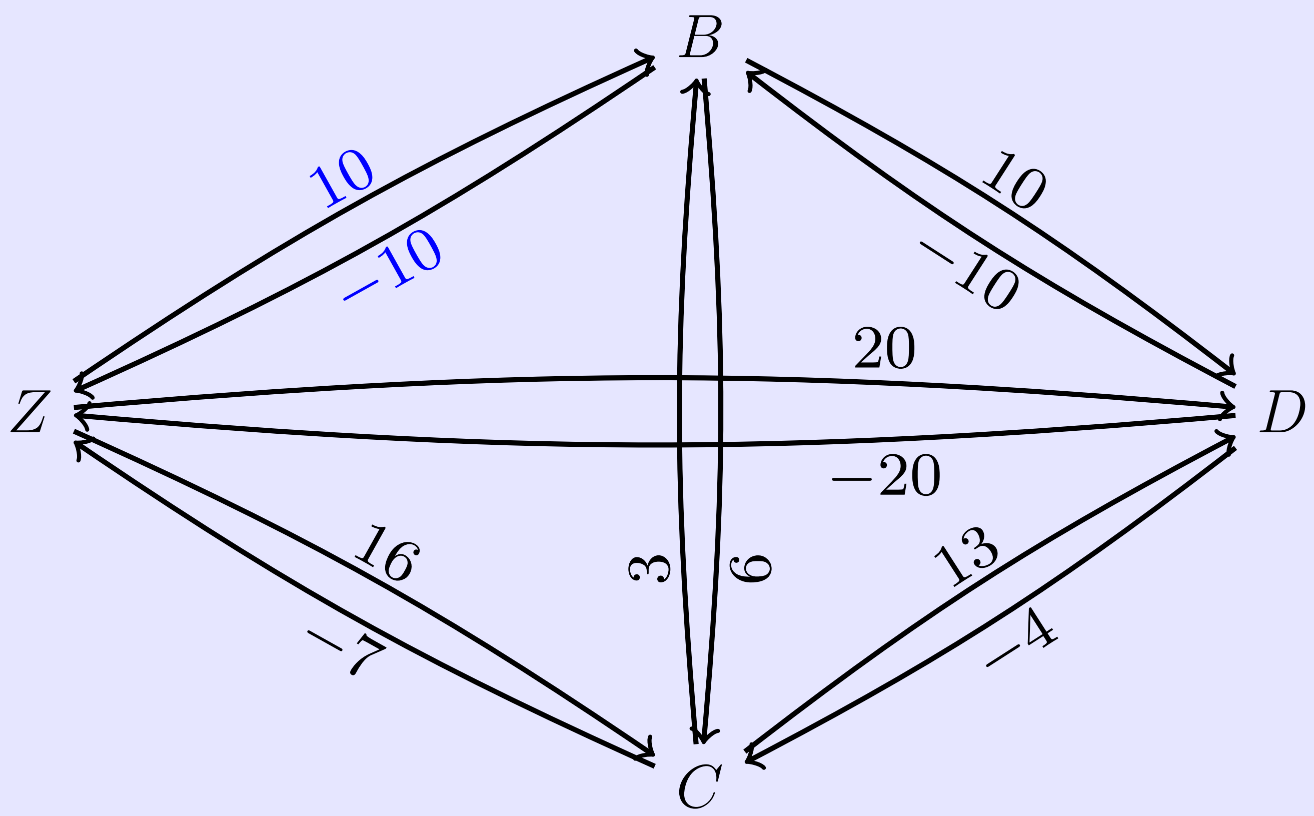
Next, select $D = 20$; and update APSP graph:



Remaining Time Windows: $B \in [5, 16],\ C \in [2, 18]$

“Solving” Sample STN (ctd.)

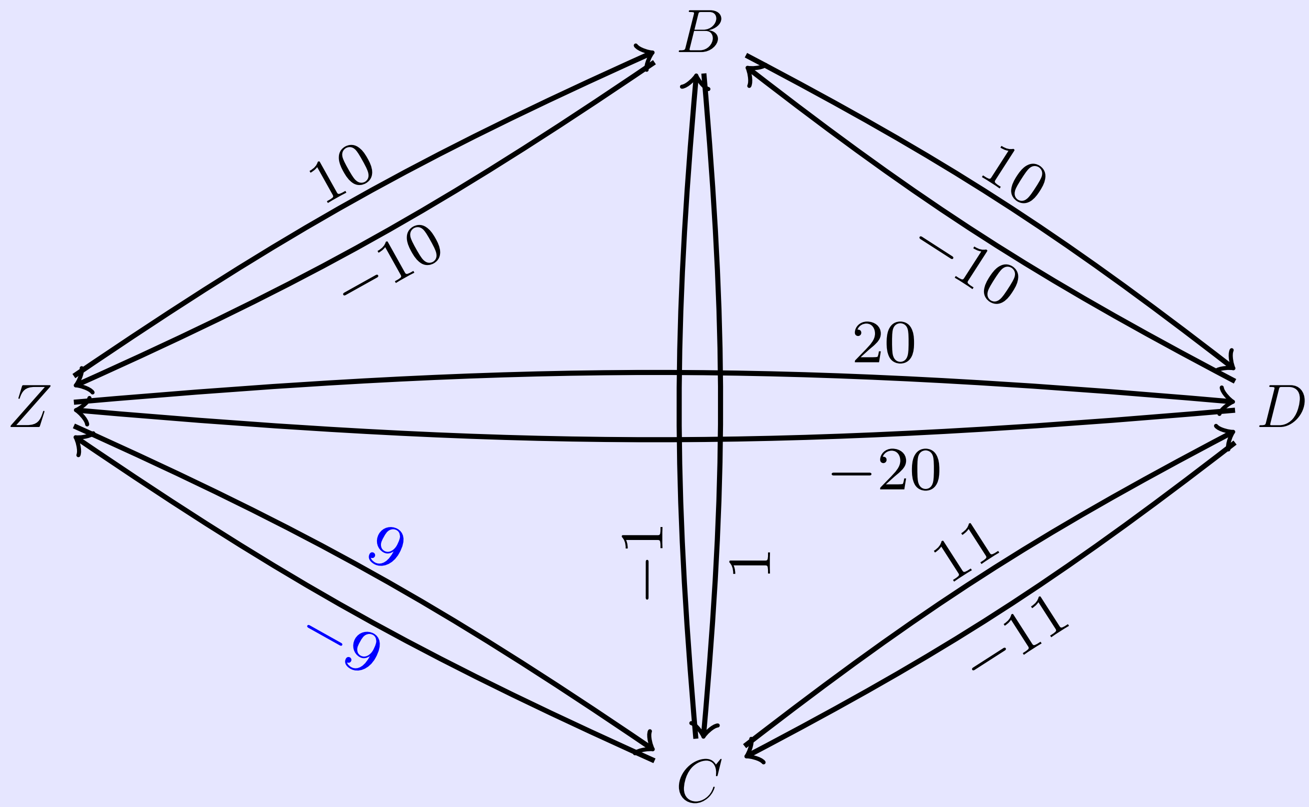
Next, select $B = 10$; and update APSP graph:



Remaining Time Windows: $C \in [7, 16]$

“Solving” Sample STN (ctd.)

Finally, select $C = 9$; and update APSP graph:



Easy to verify that this is a solution.

Problems with "Solving" an STN

- May need to go back in time:
  Pick $D = 20$, then after updating, pick $B = 10$
  (i.e., no relationship to real-time execution)

- Expensive to update $\mathcal{D}$

Executing an STN in real time

- Only executed *enabled* time-points: those having no negative edges to unexecuted time-points.

- Focus updating on entries involving Z: reduces cost to linear time per update, $O(n^2)$ overall.*

- Alternatively, prior to execution, transform STN into *dispatachable* form in $O(n^2 \log n + nm)$ time; then during execution, only need to propagate bounds to *neighboring* time-points.†

*(Hunsberger 2008); †(Muscettola, Morris, and Tsamardinos 1998), †(Tsamardinos, Muscettola, and Morris 1998)

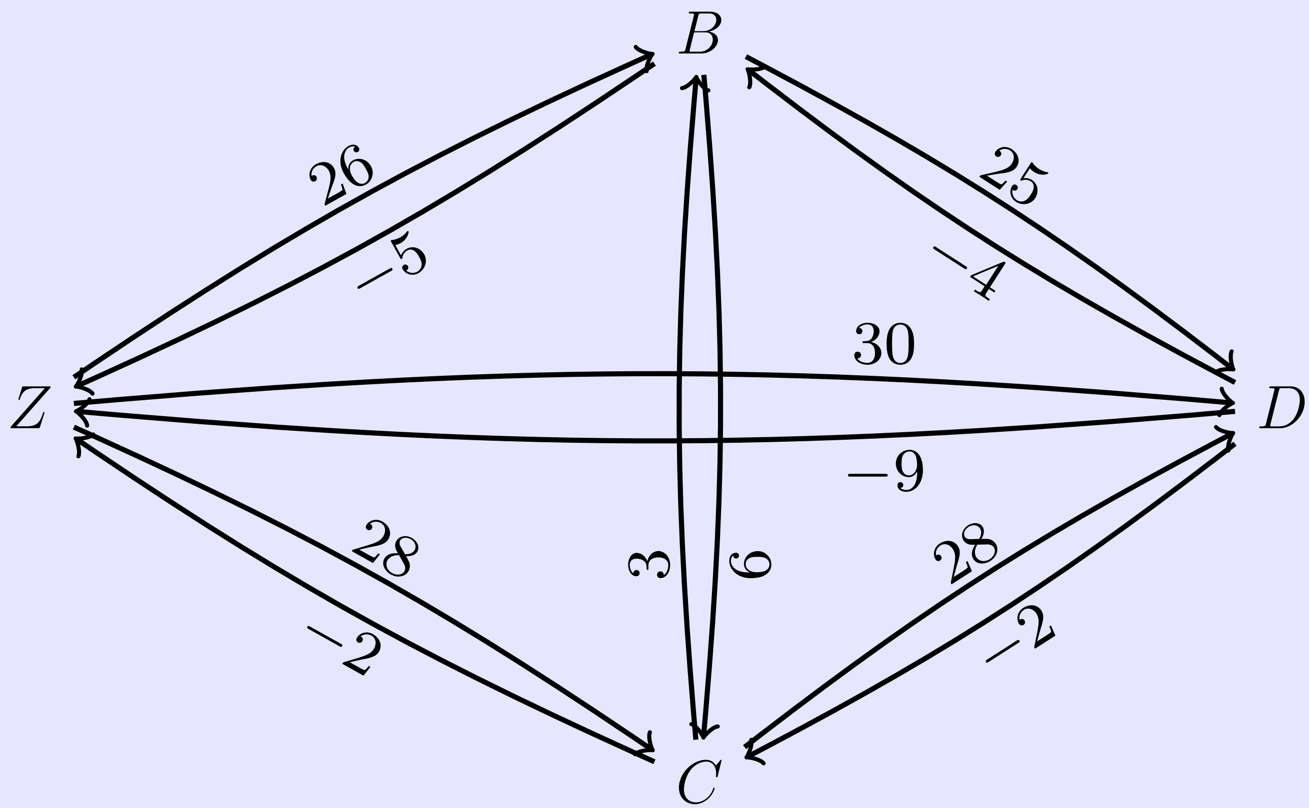
Dispatchable STN

An STN $\mathcal{S}$ is *dispatchable* if the following algorithm necessarily successfully executes $\mathcal{S}$:

1. Let $t = 0$ (current time); $\mathcal{X} = \{\}$ (executed);
   $\mathbf{E} = \{Z\}$ (currently enabled);
2. Pick any $X \in \mathbf{E}$ such that t is in X's time window;
3. Set $X := t$, and add X to $\mathcal{X}$;
4. Propagate $t \leq X \leq t$ to X's *immediate neighbors*;
5. Put into $\mathbf{E}$ all time-points Y such that all negative edges emanating from Y have a destination in $\mathcal{X}$;
6. Wait until t has advanced to some time between $\min\{lb(W) \mid W \in \mathbf{E}\}$ and $\min\{ub(W) \mid W \in \mathbf{E}\}$;
7. Repeat until all time-points are in $\mathcal{X}$ (executed).

Making STN Dispatchable

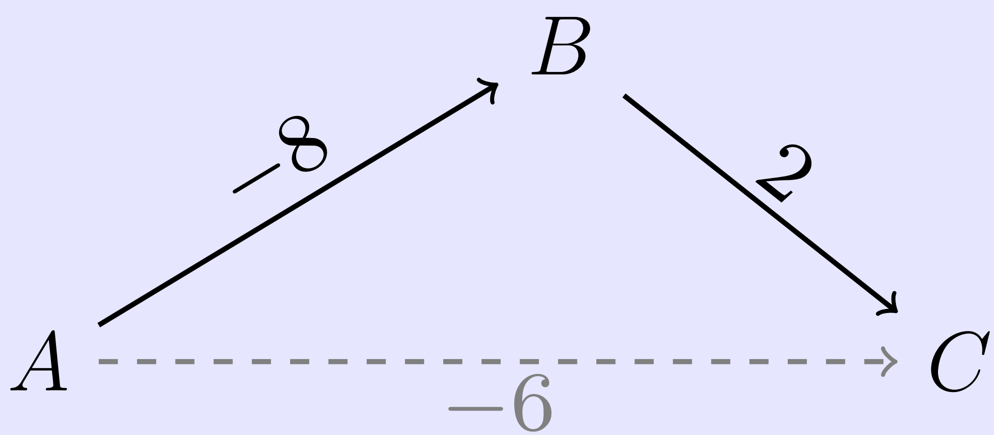
Start with APSP Graph:



Then remove *dominated* edges …

Remove Dominated Edges

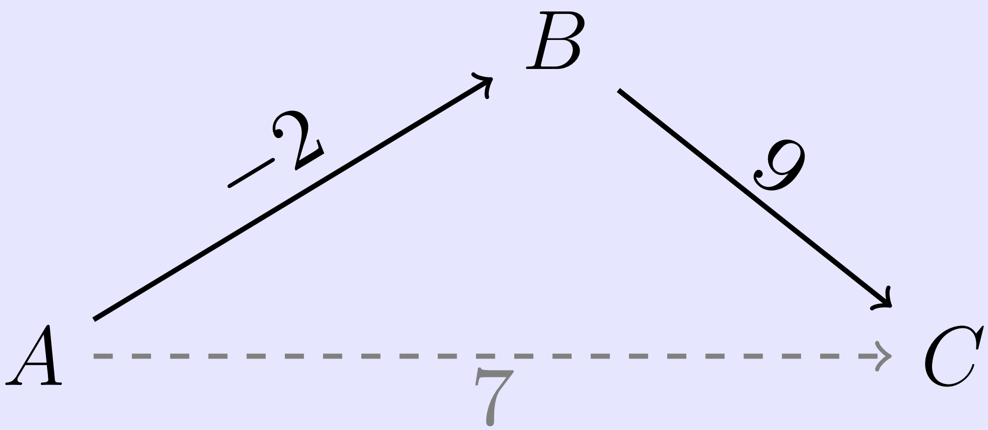
A negative edge AC is dominated by a negative edge AB if $\mathcal{D}(A,B) + \mathcal{D}(B,C) = \mathcal{D}(A,B)$:



Note: AB and AC have the *same source* node: A.

Remove Dominated Edges (ctd.)

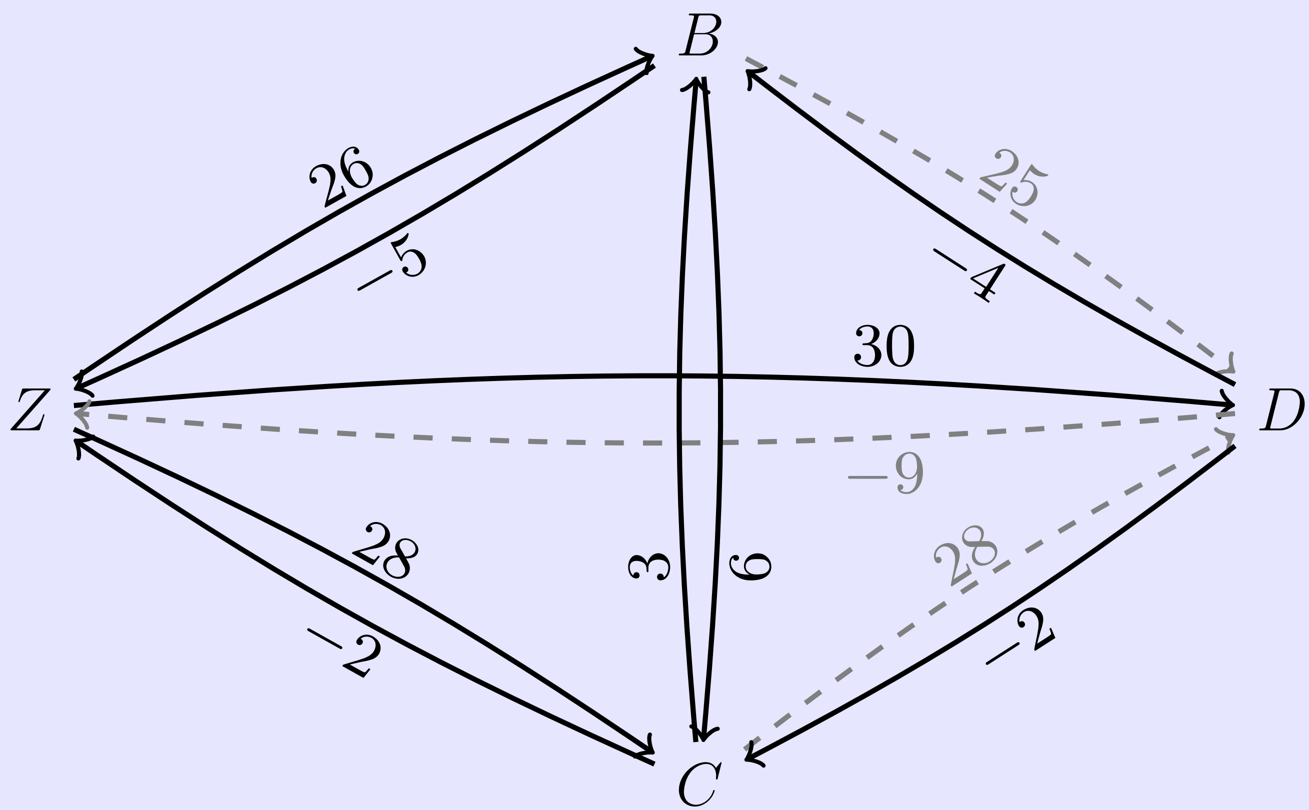
A non-negative edge AC is dominated
by a non-negative edge BC
if $\mathcal{D}(A,B) + \mathcal{D}(B,C) = \mathcal{D}(A,B)$:



Note: BC and AC have the *same destination* node: A.

Making STN Dispatchable (ctd.)

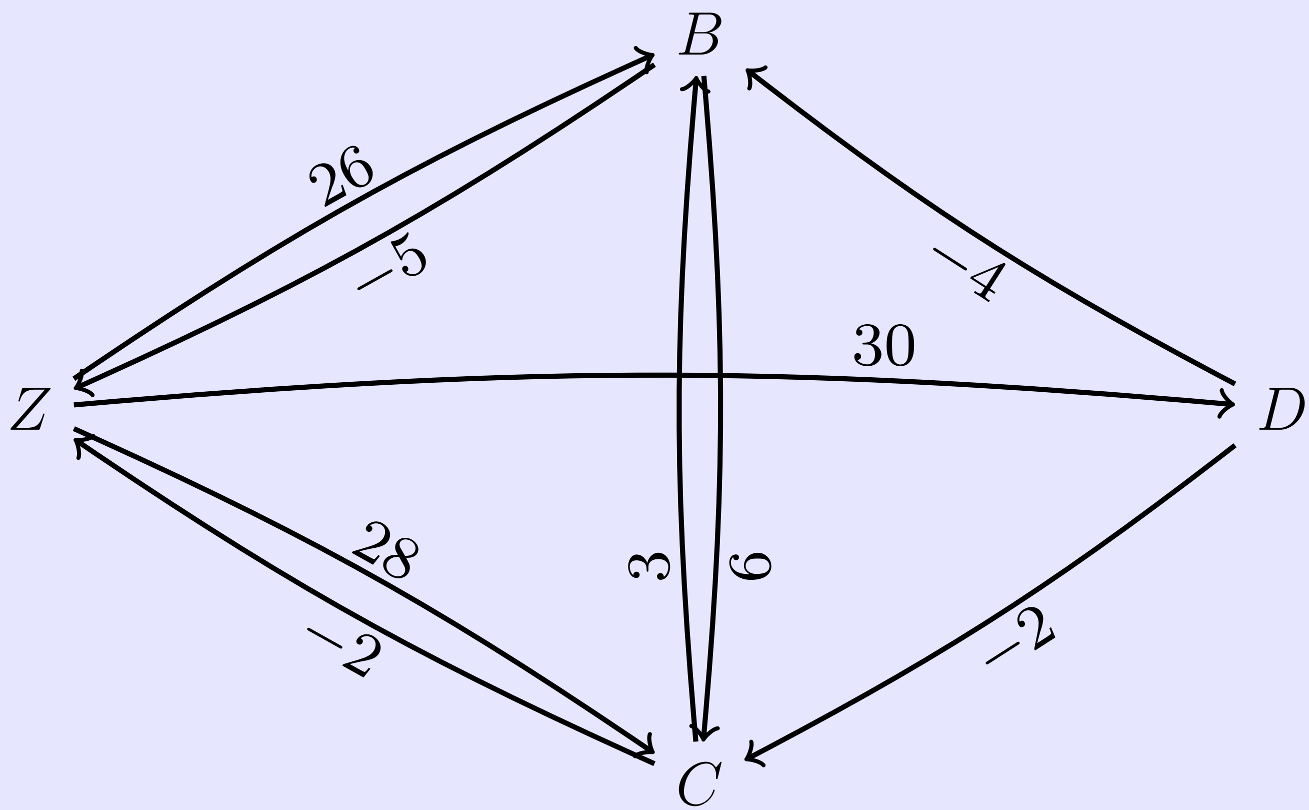
Remove "dominated" edges:*



*(Muscettola, Morris, and Tsamardinos 1998)

Dispatching the STN

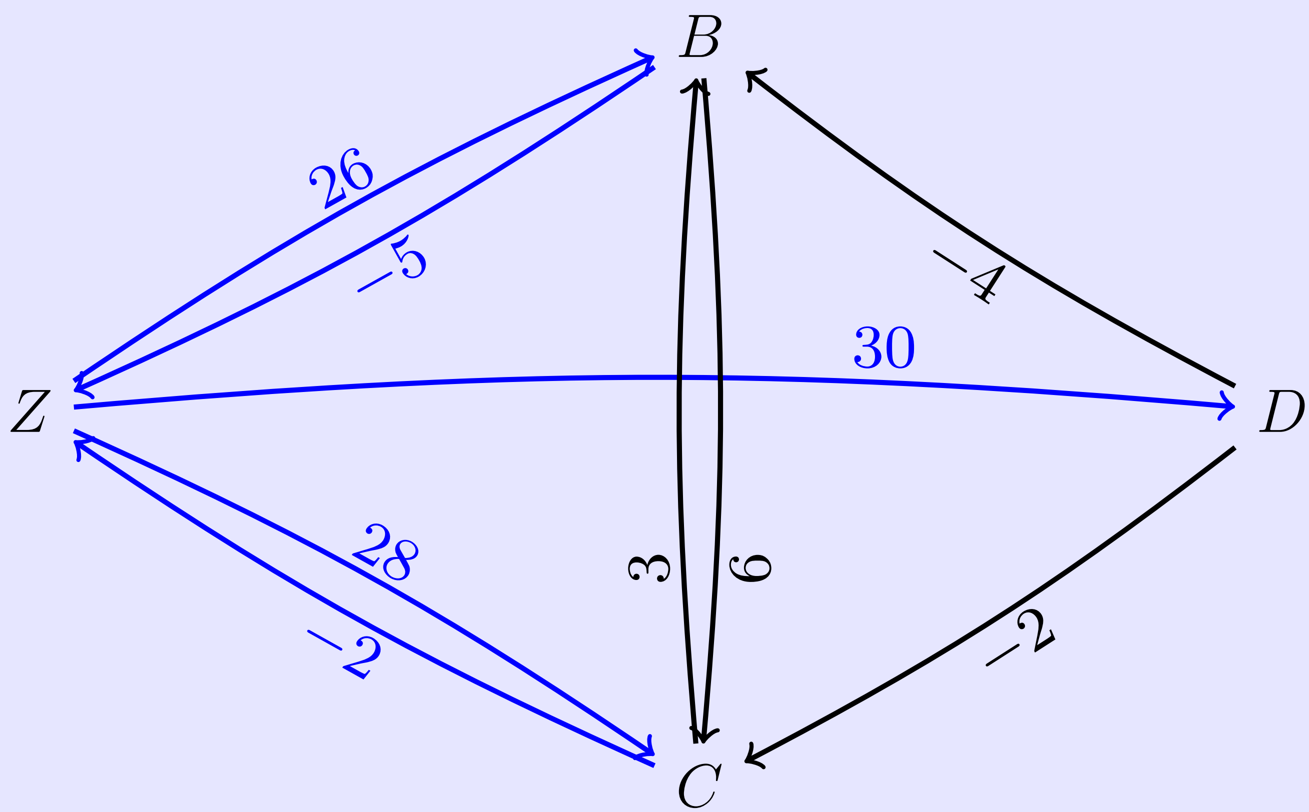
Initially: $t = 0$, $\mathcal{X} = \{\}$, $\mathbf{E} = \{Z\}$.



Pick Z from $\mathbf{E}$. Set $Z = 0$.

Dispatching the STN (ctd.)

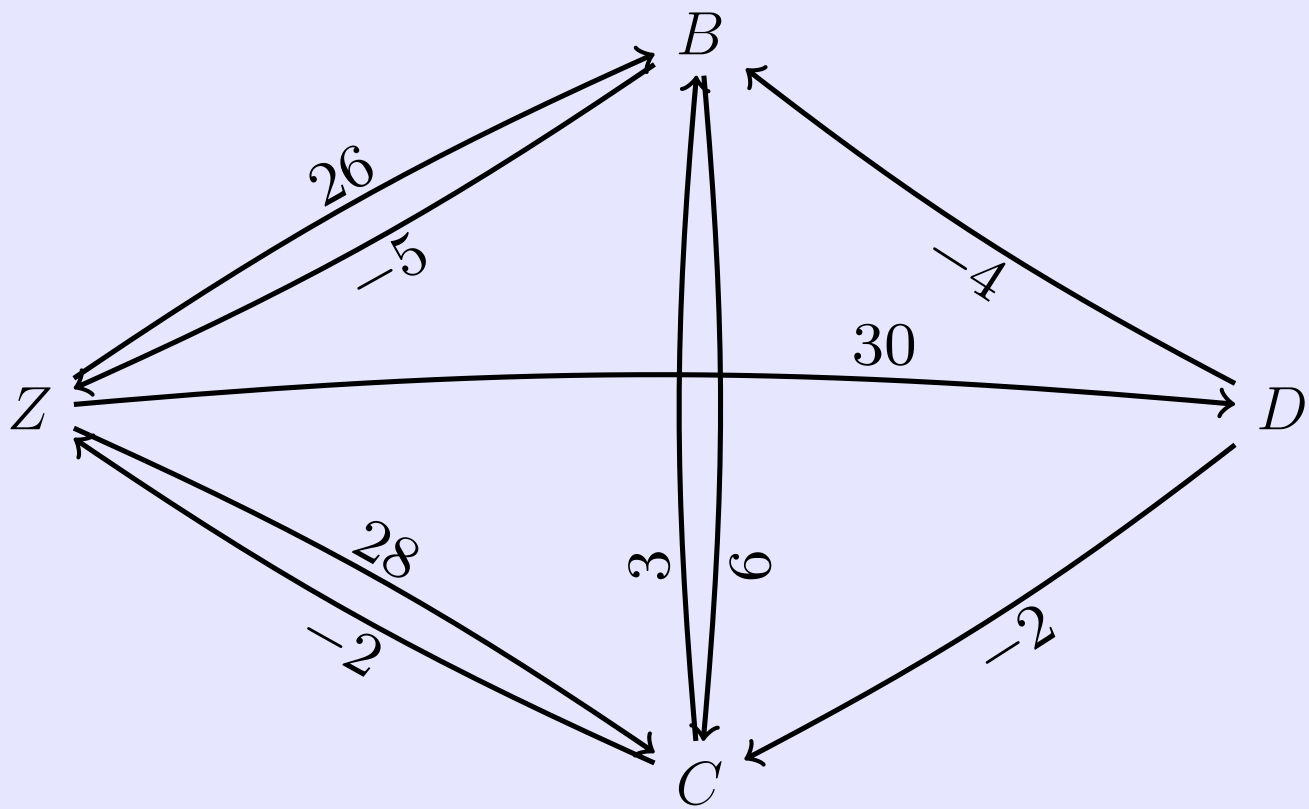
Propagate $Z = 0$ to neighbors;



$$\mathcal{X} = \{Z\}, \mathbf{E} = \{B, C\}; B \in [5, 26], C \in [2, 28], D \in [0, 30].$$

Dispatching the STN (ctd.)

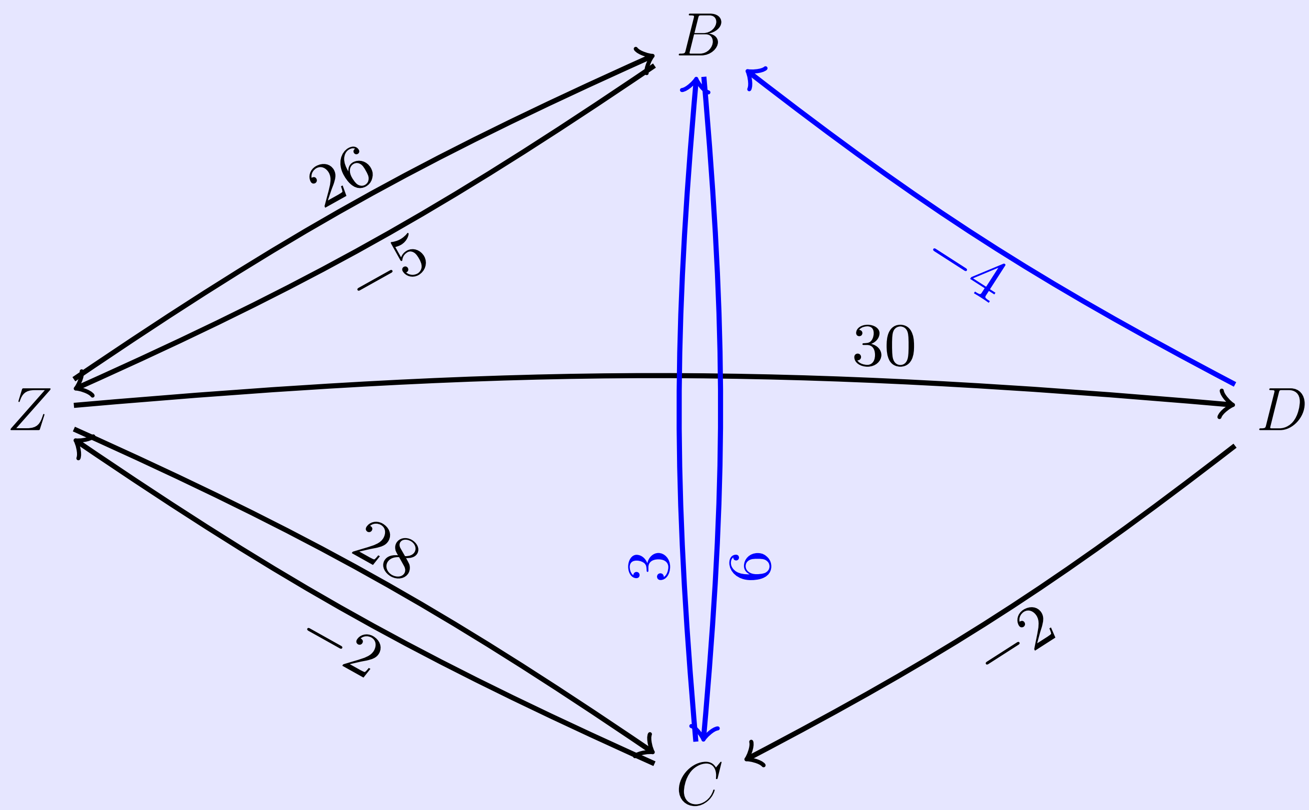
$\mathcal{X} = \{Z\}$, $\mathbf{E} = \{B, C\}$; Bounds: $B \in [5, 26]$, $C \in [2, 28]$.



Let t advance to 12; Pick B from $\mathbf{E}$; Set $B = 12$.

Dispatching the STN (ctd.)

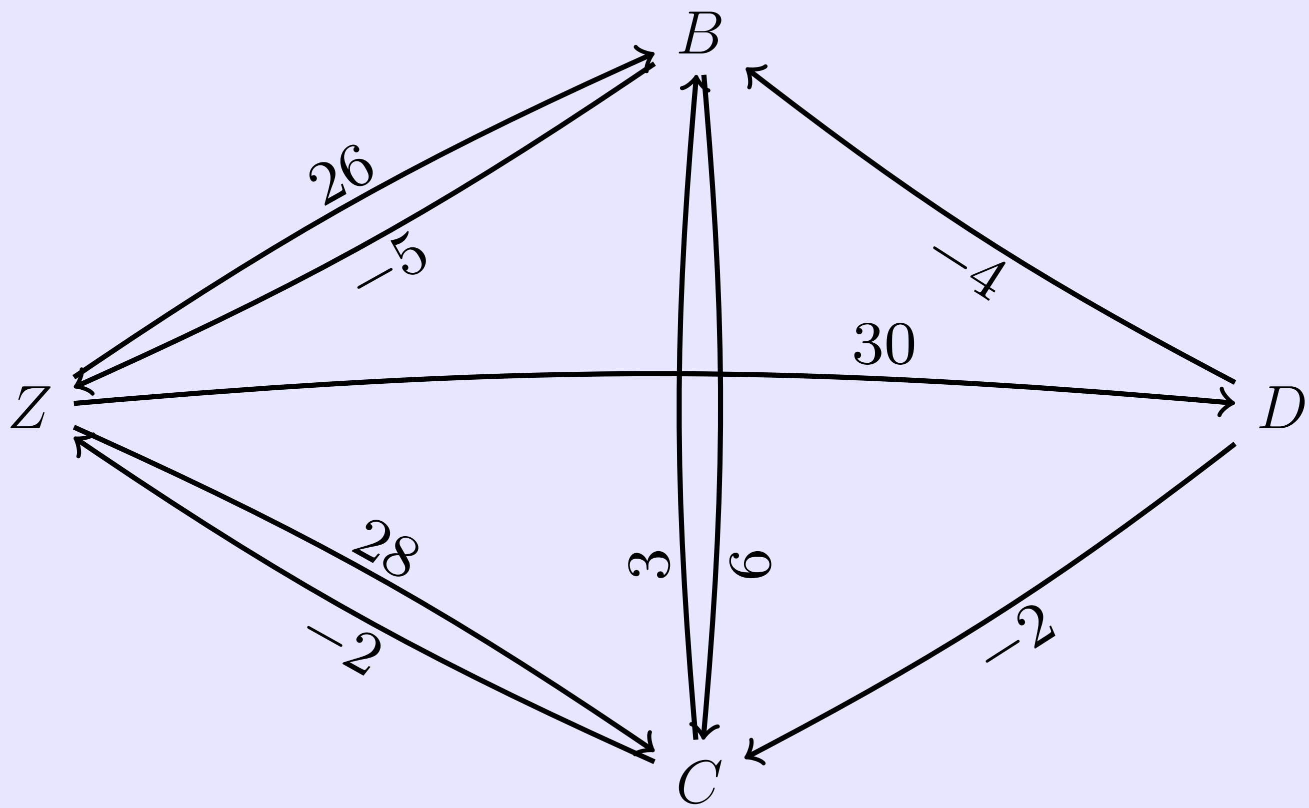
Propagate $B = 12$ to neighbors



$\mathcal{X} = \{Z, B\}, t = 12, \mathbf{E} = \{C\}, C \in [12, 18], D \in [16, 30]$

Dispatching the STN (ctd.)

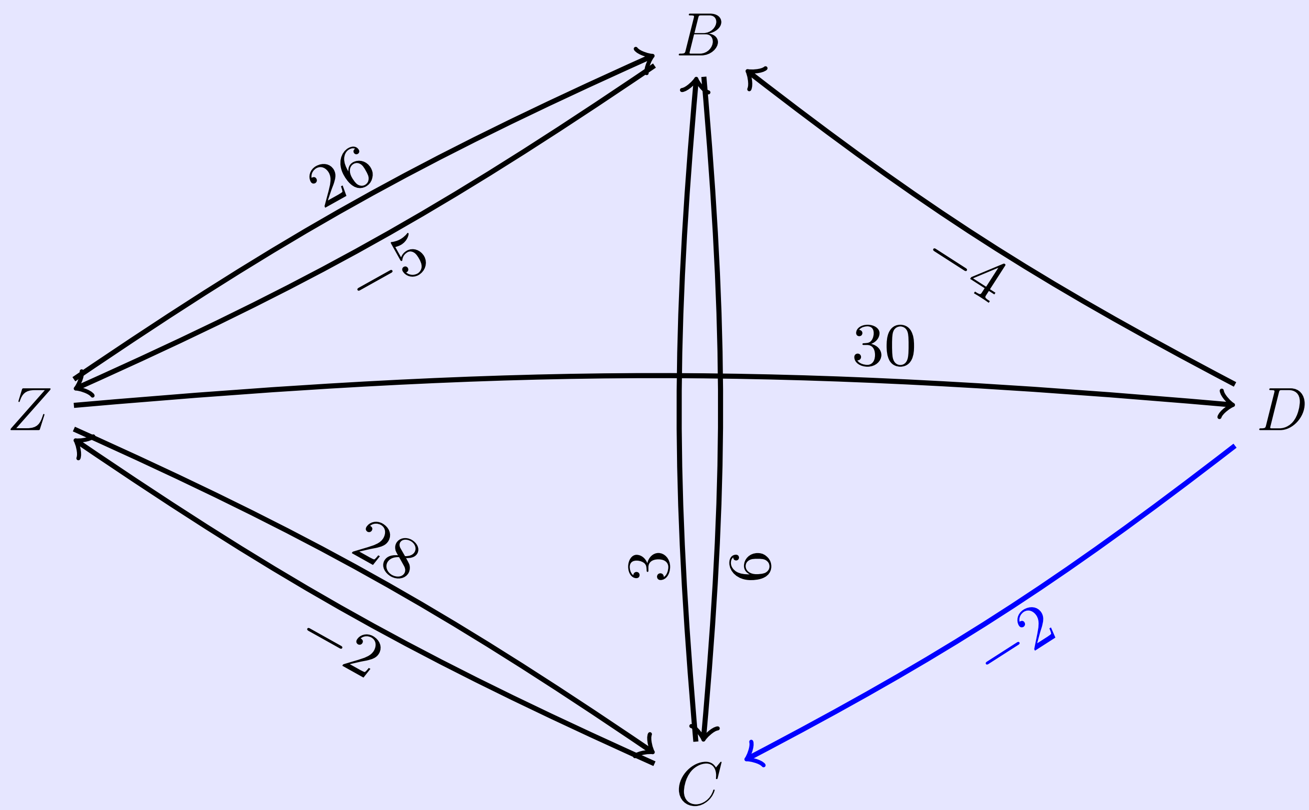
$\mathcal{X} = \{Z, B\}, t = 12, \mathbf{E} = \{C\}, C \in [12, 18], D \in [16, 30]$



Let t advance to 16, pick C from $\mathbf{E}$, set $C = 16$.

Dispatching the STN (ctd.)

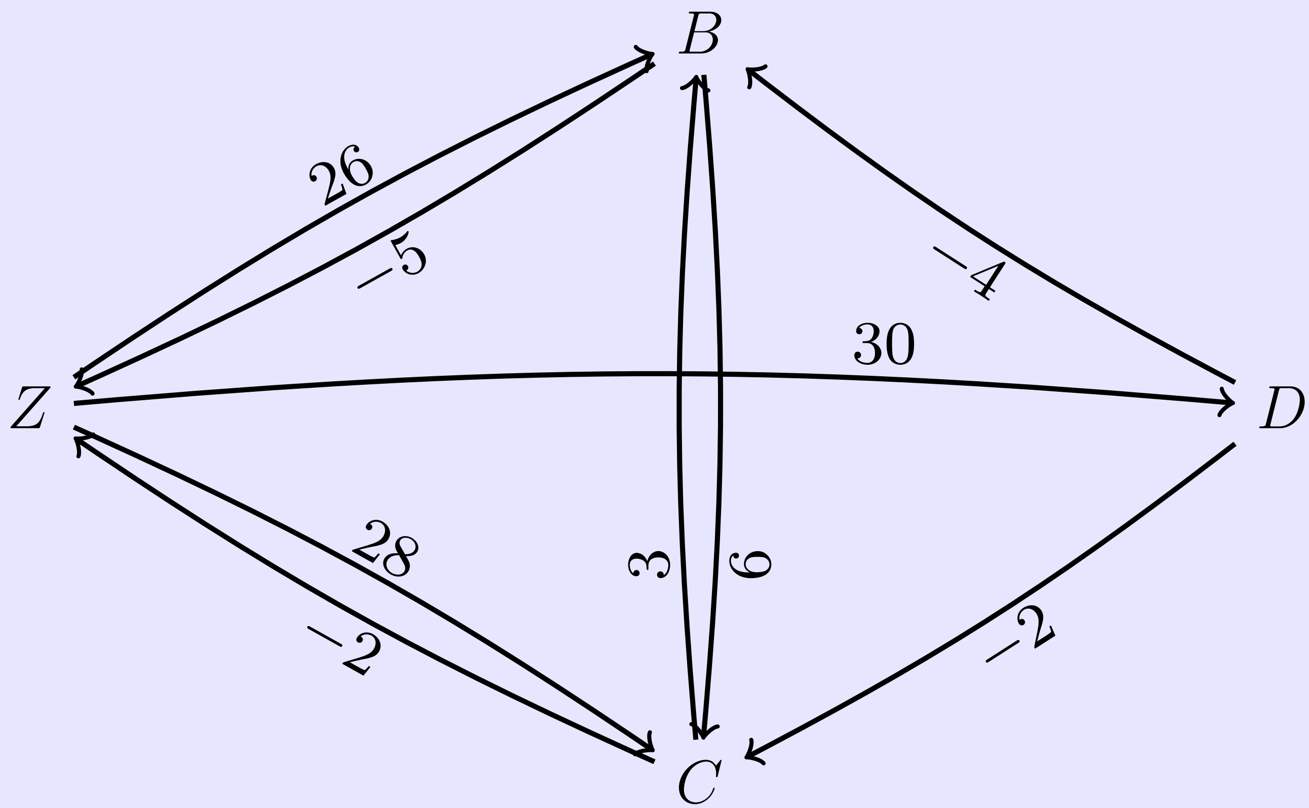
Propagate $C = 16$ to C's only remaining neighbor, D.



$$\mathcal{X} = \{Z, B, C\}, t = 16, \mathbf{E} = \{D\}, D \in [18, 30]$$

Dispatching the STN (ctd.)

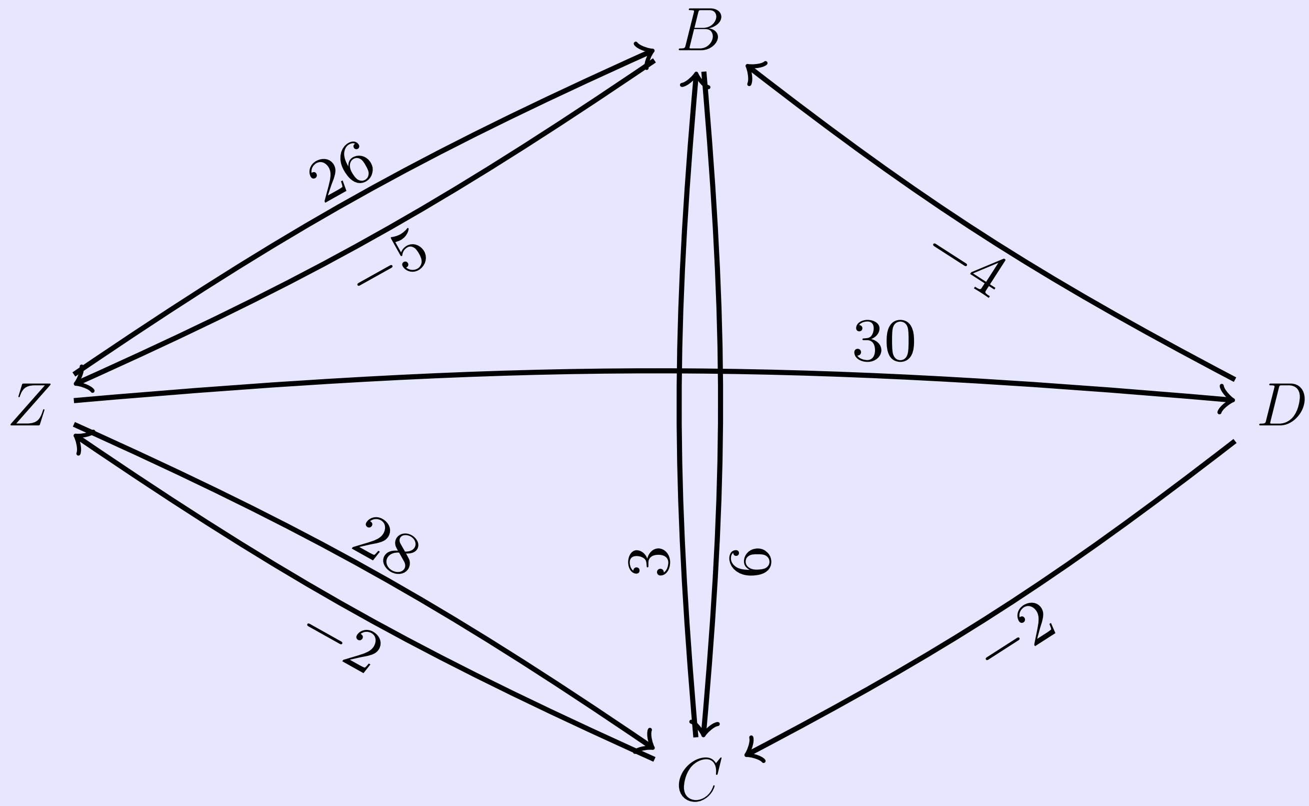
$\mathcal{X} = \{Z, B, C\}$, $t = 16$, $\mathbf{E} = \{D\}$, $D \in [18, 30]$



Let t advance to 25, pick D from $\mathbf{E}$, set $D = 25$.

Dispatching the STN (ctd.)

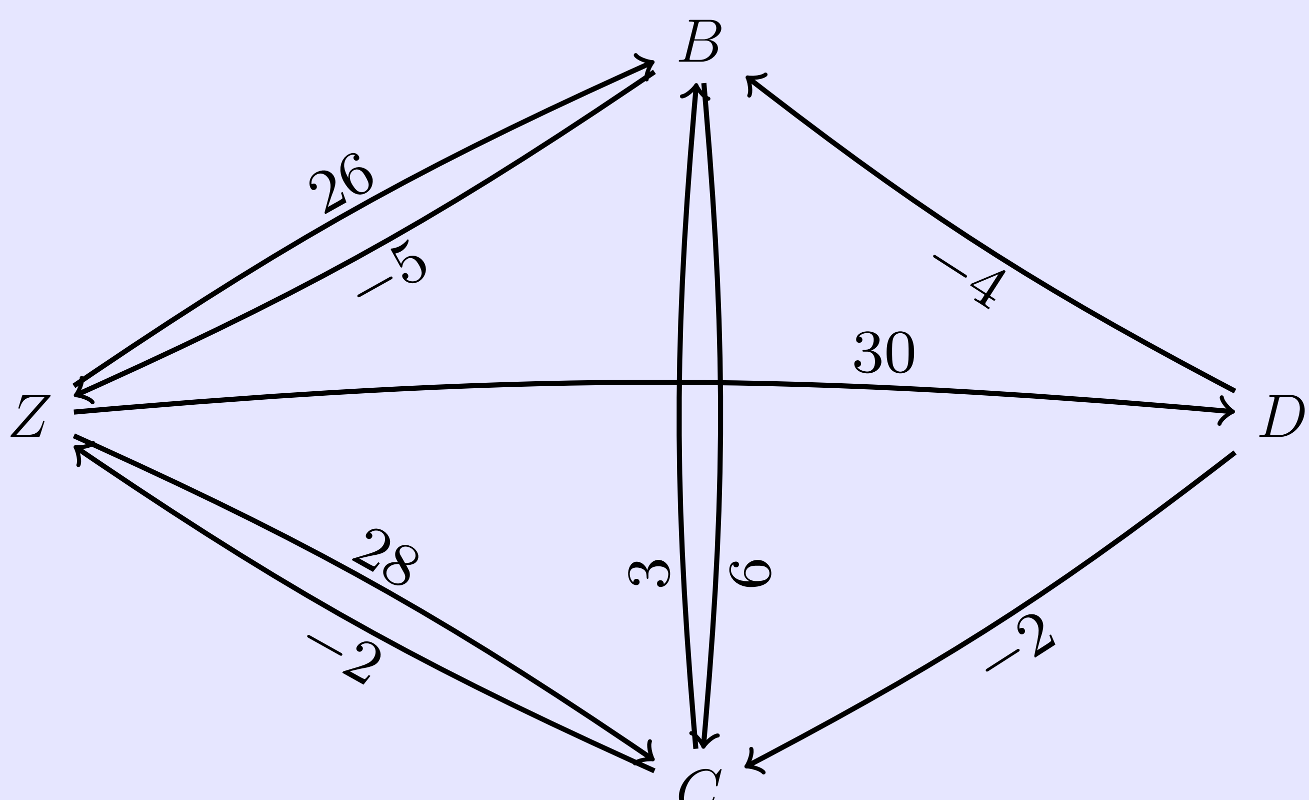
$$\mathcal{X} = \{Z, B, C, D\}, t = 25, \mathbf{E} = \{\}$$



Solution: $Z = 0, B = 12, C = 16, D = 25.$

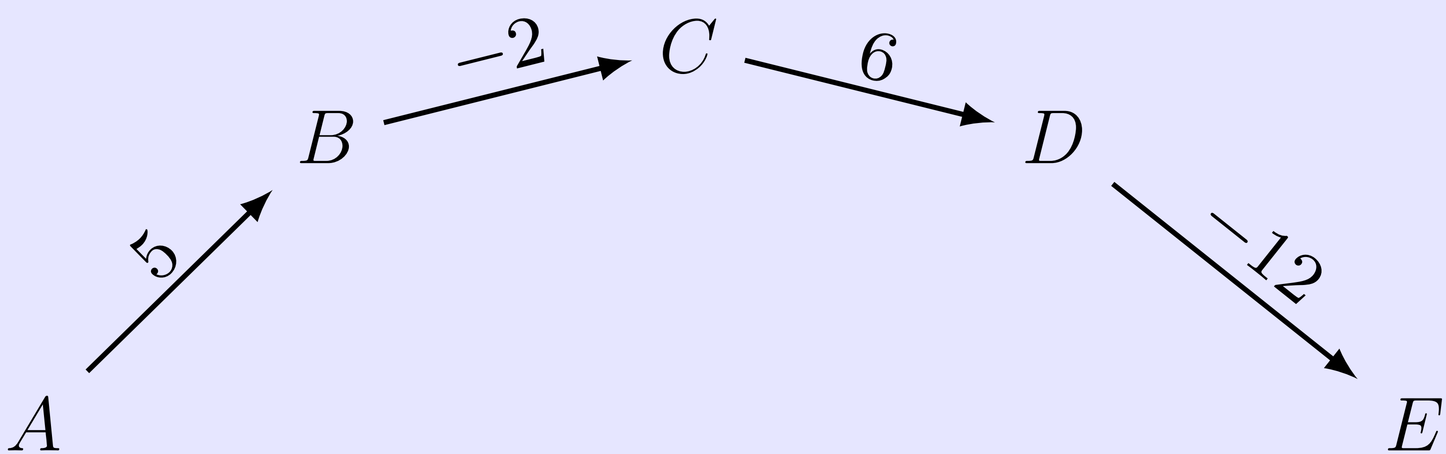
Dispatching the STN (ctd.)

Easy to check that $Z = 0, C = 20, B = 23, D = 28$ can also be generated by the dispatcher.



New View of Dispatchability*

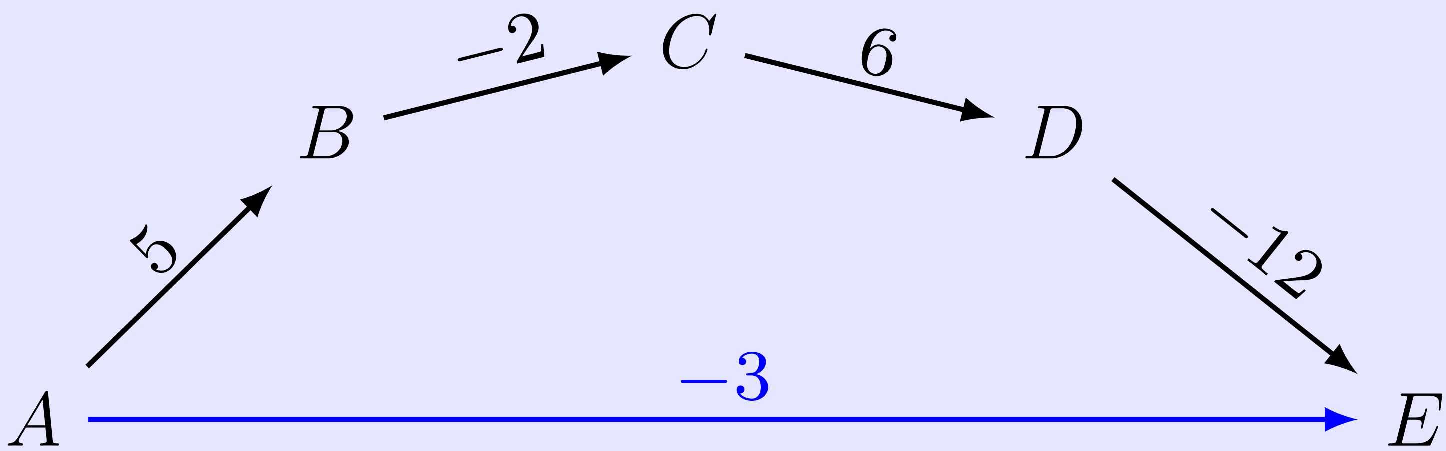
(1) A path $\mathcal{P}$ has the prefix/postfix property if every proper prefix of $\mathcal{P}$ has non-negative length, and every proper postfix of $\mathcal{P}$ has negative length.



* (Morris 2014)

New View of Dispatchability (ctd)

(2) An STN is PP-complete if for each shortest path from any A to any B that has the prefix/postfix property, there is an edge from A to B with the same length.



(3) A consistent and PP-complete STN is dispatchable.

More on Dispatchability

Morris (Morris 2016) presented further graphical analyses of the dispatchability of STNs.

STN Summary

- STNs have been used to provide flexible planning and scheduling systems for more than a decade.
- Efficient algorithms for checking consistency, incrementally updating the APSP matrix, and managing execution in real time for maximum flexibility.
- However, STNs cannot represent uncertainty (e.g., actions with uncertain durations) or conditional constraints (e.g., only do X if test result is negative).

STNs with Uncertainty

Motivation for STNUs

- You may control when an action starts, but not how long it takes to complete: taxi ride, bus ride, baseball game, medical procedure.
- Although their durations may be uncertain, they are often within known bounds.
- Such actions can be represented by *contingent links* in a temporal network …

STN with Uncertainty*

An STNU is a triple, $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ where:

- $\mathcal{T}$ and $\mathcal{C}$ as in an STN
- $\mathcal{L}$ — Contingent Links: (A, ℓ, u, C)
  - A is the activation time-point.
  - C is the contingent time-point.
  - Duration bounded: $C - A \in [\ell, u]$

    — but *uncontrollable*

* (Morris, Muscettola, and Vidal 2001)

STNU Graph

- Nodes and Edges as in an STN graph

$$Y - X \in [3, 7] \qquad \Longleftrightarrow \qquad \begin{tikzcd} X \arrow[r, bend left, "7"] & Y \arrow[l, bend left, "-3"] \end{tikzcd}$$

- Contingent Links $\Longleftrightarrow$ Labeled Edges*

$$C - A \in [3, 7] \qquad \Longleftrightarrow \qquad \begin{tikzcd} A \arrow[r, bend left, "c\,:\,3"] & C \arrow[l, bend left, "C\,:\,-7"] \end{tikzcd}$$

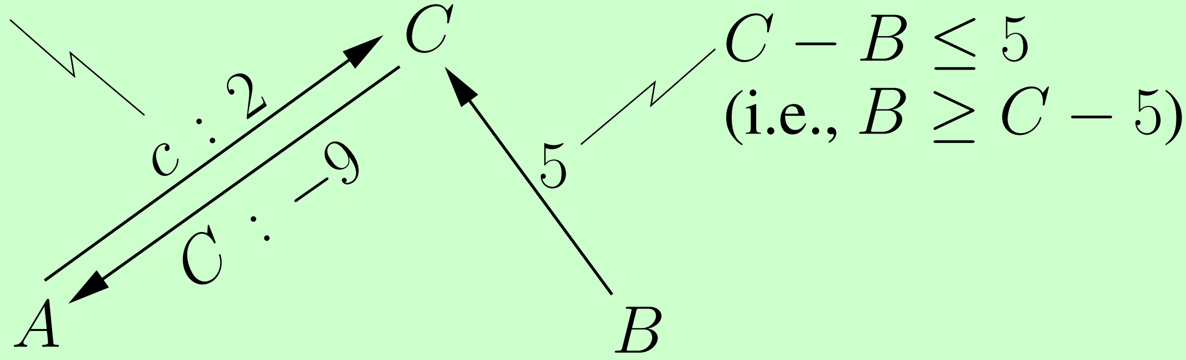
Labeled edges represent uncontrollable possibilities.

* (Morris and Muscettola 2005)

STNU Example

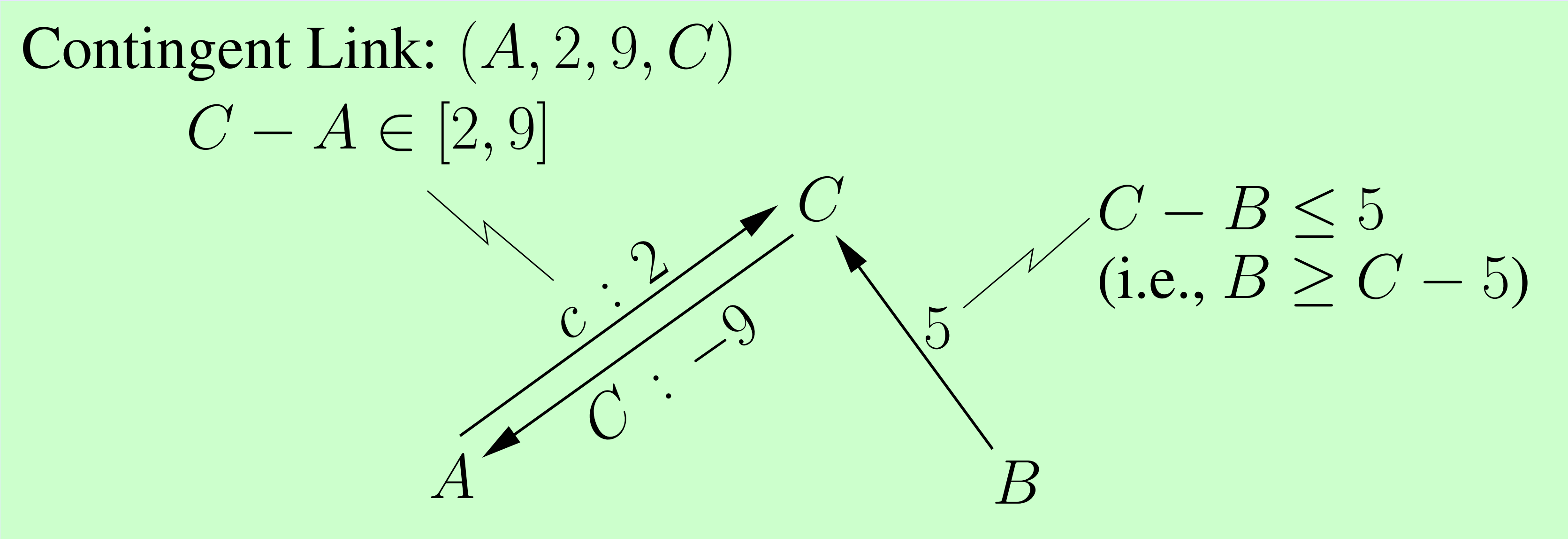
Contingent Link: $(A, 2, 9, C)$

$C - A \in [2, 9]$



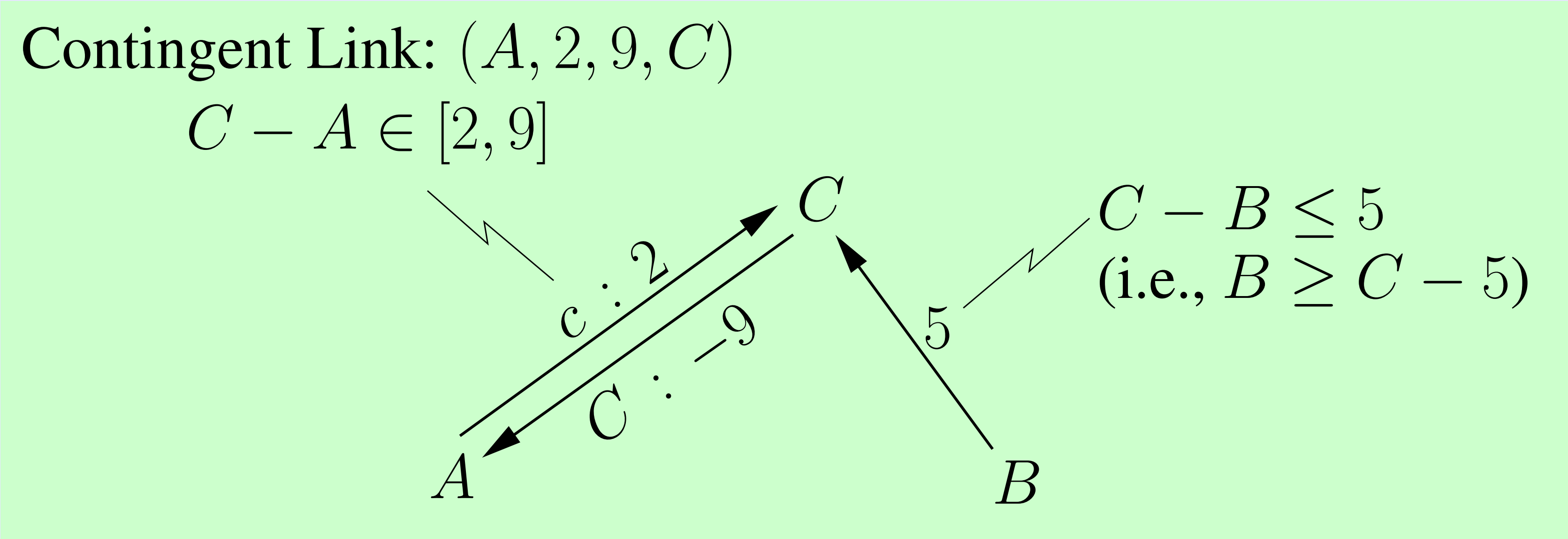
If $A = 0$, when is it safe to execute B?

STNU Example



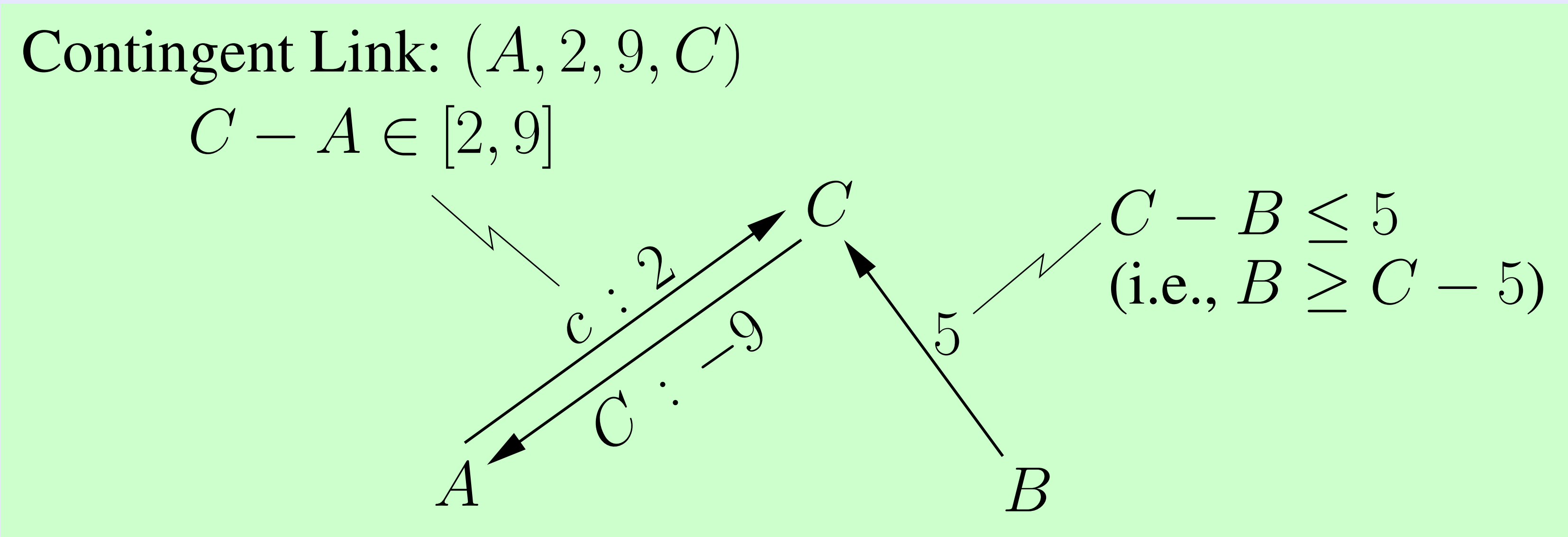
If $A = 0$ and $B = 2$, then problem if $C > 7$.

STNU Example



If $A = 0$ and $B \geq 4$, then no problems!

STNU Example



If $A = 0$ and $C = 3$, then $B > 3$ no problem!

Dynamic Controllability (DC)

An STNU is *dynamically controllable* (DC) if:

there exists a *dynamic strategy* ...

for executing the *non-contingent* time-points ...

such that *all* of the constraints will be satisfied ...

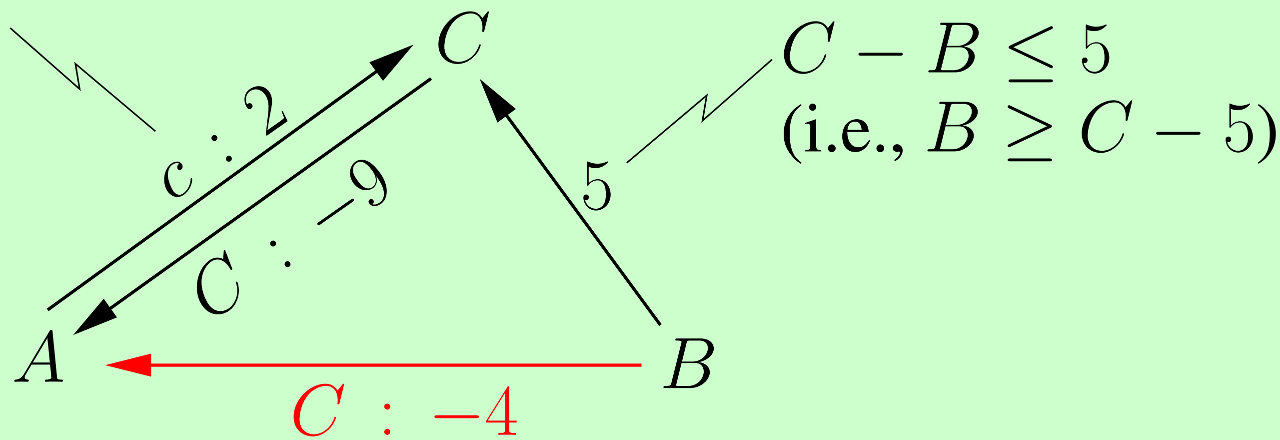
no matter how the contingent durations turn out.

$\Rightarrow$ A dynamic strategy can *react* to contingent executions.

STNU Example

Contingent Link: $(A, 2, 9, C)$

$C - A \in [2, 9]$



Strategy: As long as C unexecuted,
B must wait at least 4 after A.

Semi-Reducible Paths

- Whereas shortest paths in an STN graph represent the strongest constraints that a consistent execution must satisfy, the shortest *semi-reducible* paths in an STNU graph represent the strongest constraints that an execution strategy for an STNU must satisfy.
- The *All-Pairs, Shortest Semi-Reducible Paths* (APSSRP) matrix $\mathcal{D}^*$ for an STNU is analogous to the APSP matrix for an STN.

Fundamental Theorem of STNUs

For an STNU $\mathcal{S}$, with graph $\mathcal{G}$, and APSSRP matrix $\mathcal{D}^*$, the following are equivalent:

- $\mathcal{S}$ is dynamically controllable
- $\mathcal{G}$ has no *semi-reducible* negative loops
- $\mathcal{D}^*$ has non-negative values on its main diagonal

(Morris and Muscettola 2005; Morris 2006; Hunsberger 2010; 2013b)

DC Checking for STNUs

The worst-case time for DC-checking algorithms for STNUs has improved dramatically in recent years:

- Pseudo-polynomial: (Morris et al., 2001)
- $O(N^5)$: (Morris and Muscettola 2005)
- $O(N^4)$: (Morris 2006)
- $O(N^3)$: (Morris 2014)

And *flexibly* executing a DC STNU can be done in $O(N^3)$ time overall (Hunsberger 2013a; 2015) (Morris 2014).

Real-Time Execution Decisions*

The semantics for dynamic controllability can be stated in terms of *Real-Time Execution Decisions* (RTEDs):

- WAIT:
  Wait for some activated contingent link to complete.

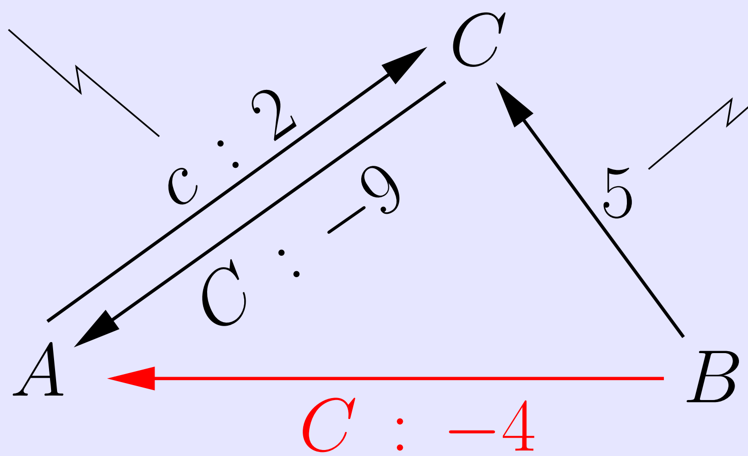
- (t, χ):
  If nothing happens before time $t \in \mathbb{R}$, then execute the (non-contingent) time-points in χ at time t.

* (Hunsberger 2009)

RTED Example

Contingent Link: $(A, 2, 9, C)$

$C - A \in [2, 9]$



$C - B \leq 5$

(i.e., $B \geq C - 5$)

Initial Decision: $(4, \{B\})$

(If nothing happens before time 4, execute B at 4.)

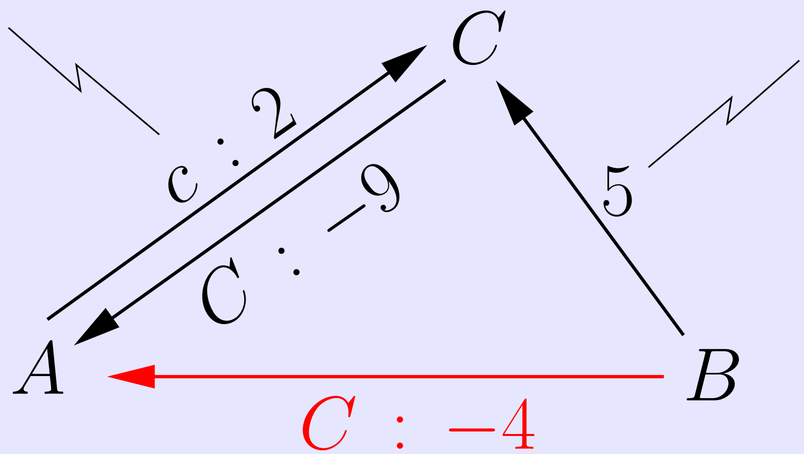
RTED Example (ctd.)

Contingent Link: $(A, 2, 9, C)$

$C - A \in [2, 9]$

$C - B \leq 5$

(i.e., $B \geq C - 5$)



Possible Outcome: C executes at time 2.

Next decision: $(3, \{B\})$

(If nothing happens before time 3, execute B at 3.)

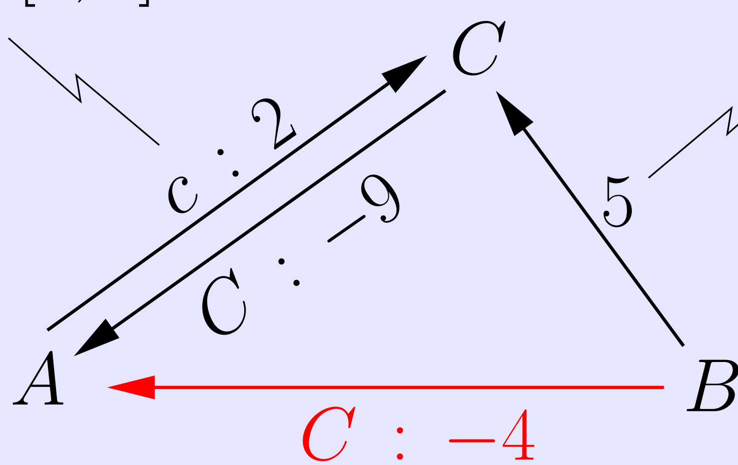
RTED Example (ctd.)

Contingent Link: $(A, 2, 9, C)$

$C - A \in [2, 9]$

$C - B \leq 5$

(i.e., $B \geq C - 5$)



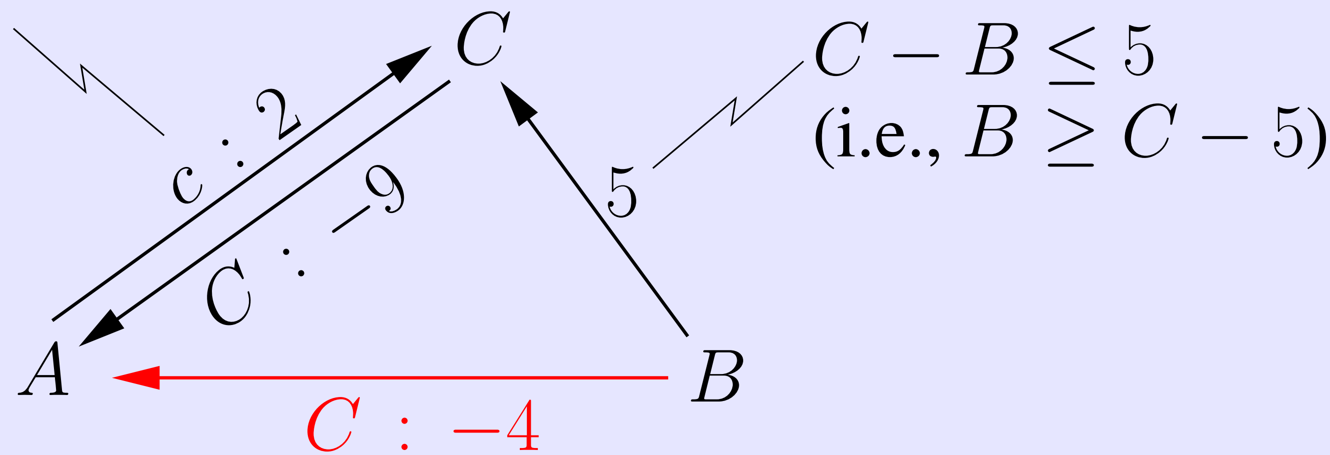
Initial Decision: $(4, \{B\})$

(If nothing happens before time 4, execute B at 4.)

RTED Example (ctd.)

Contingent Link: $(A, 2, 9, C)$

$C - A \in [2, 9]$



Possible Outcome: C does not execute yet;
so B is executed at 4

Next decision: WAIT (for C to execute)

Edge-Generation Rules

- *No Case* Rule
- *Upper-Case* Rule
- *Lower-Case* Rule
- *Cross-Case* Rule
- *Label-Removal* Rule

(Morris and Muscettola 2005)

The No-Case Rule

$$
\begin{tikzcd}[column sep=huge]
Q \arrow[r, "3"'] \arrow[rr, bend left, "\color{red}\mathbf{7}"] & S \arrow[r, "4"'] & T
\end{tikzcd}
$$

The Upper-Case Rule

$$\begin{tikzcd}[column sep=huge] Q \arrow[r, "3"'] \arrow[rr, bend left, "C\!:\!-7"] & C \arrow[r, "C\!:\!-10"'] & A \end{tikzcd}$$

The Lower-Case Rule

$$\begin{tikzcd}[column sep=huge] A \arrow[r, "c:3"'] \arrow[rr, bend left, "-2"] & C \arrow[r, "-5"'] & X \end{tikzcd}$$

(Applies since $-5 \leq 0$)

The Cross-Case Rule

$$\begin{tikzcd} A \arrow[r, "c:3"'] \arrow[rr, bend left, "C_2:-5"] & C \arrow[r, "C_2:-8"'] & A_2 \end{tikzcd}$$

(Applies since $-8 \leq 0$ and $C \not\equiv C_2$)

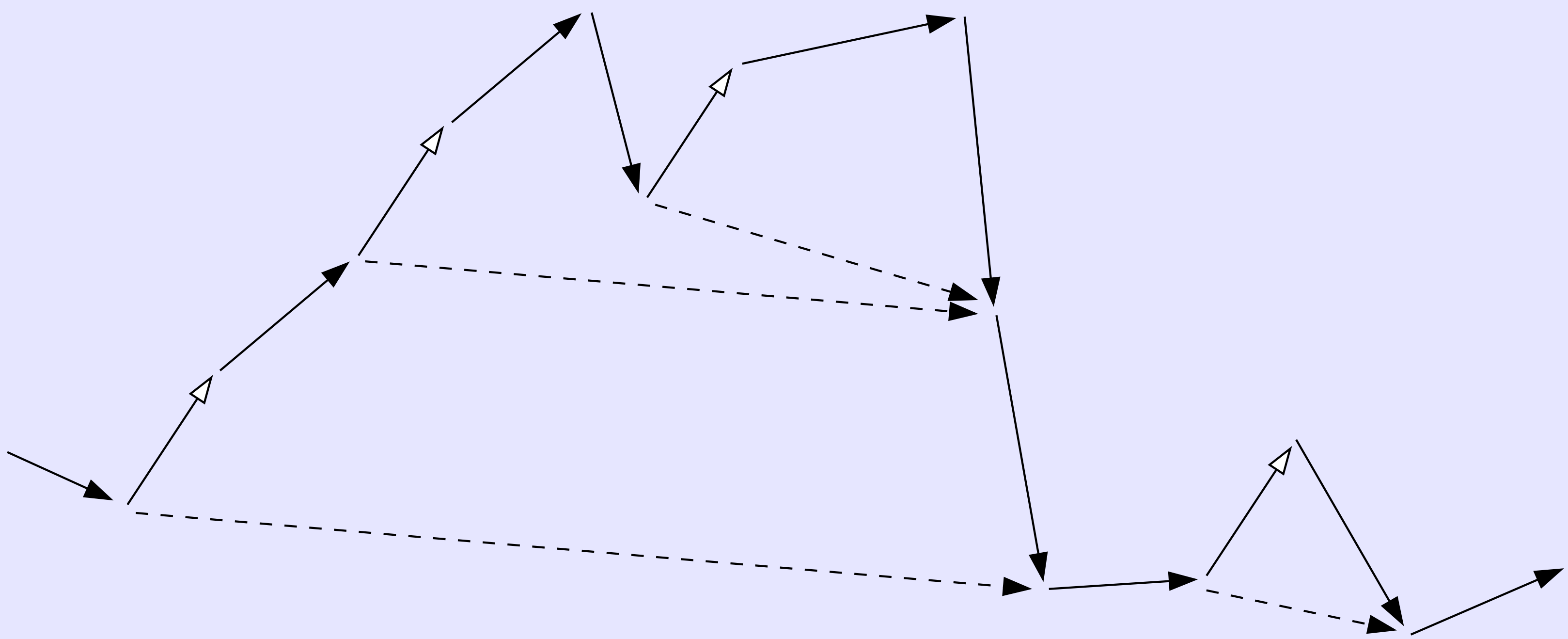
The Label-Removal Rule

$$\begin{tikzcd}[column sep=huge] X \arrow[r, "C:-1"'] \arrow[r, bend left, "-1", color=red] & A \arrow[r, "c:3"'] & C \end{tikzcd}$$

(Applies since $1 \leq 3$)

Semi-Reducibility

A path is *semi-reducible* if it can be transformed into a path with no *lower-case* edges.



Fundamental Theorem of STNUs

For an STNU $\mathcal{S}$, with graph $\mathcal{G}$, and APSSRP matrix $\mathcal{D}^*$, the following are equivalent:

- $\mathcal{S}$ is dynamically controllable
- $\mathcal{G}$ has no *semi-reducible* negative loops
- $\mathcal{D}^*$ has non-negative values on its main diagonal

(Morris and Muscettola 2005; Morris 2006; Hunsberger 2010; 2013b)

Flexible Execution of STNUs

- A DC STNU can be flexibly executed, incrementally computing updates using $O(N^2)$-time per execution event, $O(N^3)$-time overall.*

- As will be seen, this execution algorithm can be characterized as a dispatching algorithm for STNUs.

* (Hunsberger 2013a; 2015)

STNU Dispatchability

- For a DC STNU, Morris' $O(N^3)$-time DC-checking algorithm generates a dispatchable STNU.*

- Dispatchability same as for STNs, except that:
  - $\star$ contingent time-points are not controllable; and
  - $\star$ there are *wait* constraints: "As long as C unexecuted, X must wait at least 5 after A."

- Corollary: For a DC STNU, the STNU graph generated by exhaustively applying the constraint propagation rules from Morris et al. (2005) is dispatchable.

*(Morris 2014)

STNU Dispatchability (ctd.)

- Definition: A projection of an STNU is the STN that results from fixing the duration of each contingent link to one of its legal values.

- Definition: An STNU (including any wait constraints) is dispatchable if each of its STN *projections* is dispatchable (as an STN).

- Theorem: A dispatchable STNU is DC.*

* (Morris 2014)

STNU Summary

- The theory of STNUs (dynamic controllability, dispatchability, flexible execution) has been advanced dramatically over the past few years.
- Many important contributions from Paul Morris and colleagues.
- STNUs are ready for prime time!

Conditional STNs

Motivation for CSTNs

- Many actions generate information (e.g., medical tests, opening a box, monitoring traffic).

- The generated information is generally not known in advance, but discovered in real time.

- Some actions only make sense in certain scenarios (e.g., don't give drug if test result is negative).

- An execution strategy could be more flexible if it could react dynamically to generated information.

Motivation for CSTNs (ctd.)

- Many businesses using *workflow management systems* to automate manufacturing processes.
- Hospitals can use workflows to represent possible treatment pathways for a patient.
- CSTNs can serve as the temporal foundation for workflow management systems.

Conditional STNs (CSTNs)*

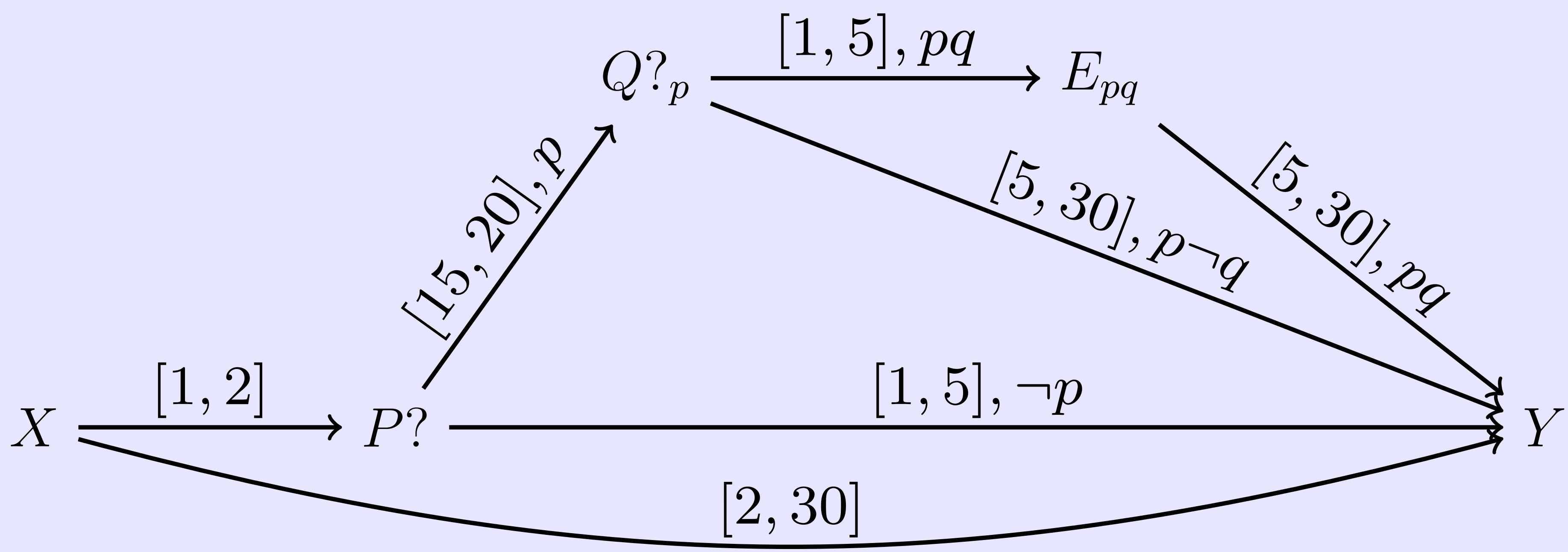
- Time-points and temporal constraints like in STNs
- Observation time-points generate truth values for propositional letters
- Time-points and constraints labeled by conjunctions of propositional letters

* (Tsamardinos, Vidal, and Pollack 2003)

Propositional Labels in CSTNs

- Propositional letters: $p, q, r, s, t, \ldots$.
- Each p has corresp. observation time-point, $P?$; executing $P?$ generates truth value for p.
- Label: conjunction of literals (e.g., $p(\neg q)r$).
- A scenario specifies values for *all* letters; the real scenario is only revealed incrementally.
- Time-points and constraints can be labeled; they only apply in scenarios where their labels are true.

Sample CSTN



$P?$ and $Q?$ represent tests for a patient.
$Q?$ is called a *child* of $P?$.

Dynamic Consistency of CSTNs

- Dynamic Execution Strategy: execution decisions may react to observations.
- A CSTN is *dynamically consistent* if there exists a dynamic execution strategy that guarantees that all *relevant* constraints will be satisfied no matter which scenario is incrementally revealed over time.

DC-Checking for CSTNs

- Convert to Disjunctive Temporal Network (Tsamardinos, Vidal, and Pollack 2003)
- Convert to Timed Game Automaton (Cimatti et al. 2014)
- Convert to Hyper Temporal Network (Comin and Rizzi 2015)
- Propagate labeled constraints (Hunsberger, Posenato, and Combi 2015)

DC Checking via Propagation

- Propagate *labeled* constraints
  — Motivated by related work (Conrad and Williams 2011)
- Introduce new kinds of literals and labels:
  Q-literals (e.g., $p?$) and *Q-labels* (e.g., $p\neg q(r?)s$)
- Address *negative q-loops* and *negative q-stars*

Labeled Constraints

$$X \xrightarrow{\quad\langle \delta, \ell \rangle\quad} Y$$

$Y - X \leq \delta$ must hold in scenarios where ℓ is true.

(If $\ell = \boxdot$, then $Y - X \leq \delta$ must hold in all scenarios.)

Propagation Rules for CSTNs

Labeled Propagation:	LP and qLP
Label Modification:	R_0 and qR_0
Label "Spreading":	R_3^* and qR_3^*

(The "q" rules propagate q-labeled constraints.)

The LP Rule

$$\begin{tikzcd} W \arrow[r, "\langle 3, pq \rangle"'] \arrow[rr, bend left, "\langle 8, pq\neg r \rangle", color=red] & X \arrow[r, "\langle 5, q\neg r \rangle"'] & Y \end{tikzcd}$$

Labels of two pre-existing edges are conjoined;
The resulting label must be consistent.

The R_0 Rule

$$\begin{tikzcd}
P? \arrow[rr, bend left, "{\langle -5, q\neg r\rangle}"] \arrow[rr, "{\langle -5, pq\neg r\rangle}"'] & & X
\end{tikzcd}$$

Edge weight must be negative;
Any occurrence of p (or $\neg p$) removed from label.

The R_3^* Rule

$$\begin{tikzcd}[column sep=huge]
P? \arrow[r, "{\langle -3, qr \rangle}"'] & X & Y \arrow[l, "{\langle -8, pqs \rangle}"'] \arrow[l, bend right, red, "{\color{red}\langle -3, qrs \rangle}"']
\end{tikzcd}$$

Pre-existing labels must be consistent;
Generated label is conjunction of pre-existing labels
— minus any occurrence of p (or $\neg p$);
Lefthand weight must be negative;
Generated weight is max of pre-existing weights.

Propagating Q-Labels

- Propagating along consistent labels insufficient
- *Q-labels*: contain literals such as $p?$. A constraint labeled by $p?$ must hold as long as p's value unknown.
- Conjunction operation expanded:
  $$p \wedge \neg p \equiv p?; \;\; p \wedge p? \equiv p?; \;\; \neg p \wedge p? \equiv p?; \;\; \text{etc.}$$
- Q-labels only needed on *lower-bound* constraints (i.e., edges pointing at Z).

The qLP Rule

$$\begin{tikzcd}[column sep=huge] W \arrow[r, "\langle -3, pq\rangle"'] \arrow[rr, bend left, "\langle -8, p(q?)\neg r\rangle"] & X \arrow[r, "\langle -5, \neg q \neg r\rangle"'] & Z \end{tikzcd}$$

Generated edge terminates at Z;
Labels need not be consistent;
Edge weights must be negative.

The qR_0 Rule

$$\begin{tikzcd} P? \arrow[rr, bend left, "{\langle -5, q\neg r\rangle}"] \arrow[rr, "{\langle -5, (p?)q\neg r\rangle}"'] & & Z \end{tikzcd}$$

Edge must terminate at Z;
Edge weight must be negative;
Any occurrence of p (or $\neg p$ or $p?$) removed from label.

The $\mathbf{qR}_3^*$ Rule

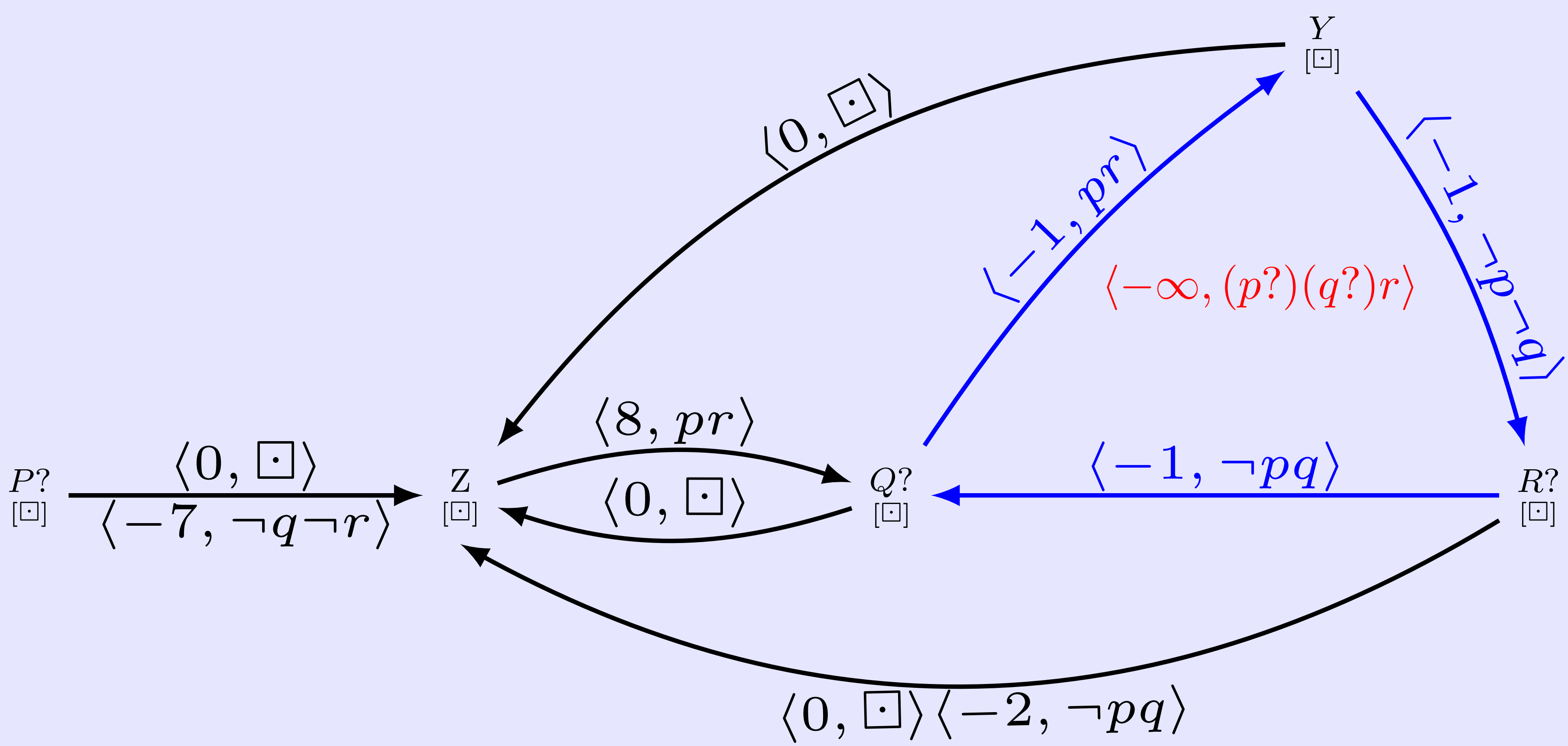
$$
\begin{tikzcd}
P? \arrow[r, "{\langle -3, q(r?) \rangle}"'] & Z & Y \arrow[l, "{\langle -8, p\neg qr(s?) \rangle}"] \arrow[l, bend right, red, "{\color{red}\langle -3, (q?)(r?)(s?) \rangle}"']
\end{tikzcd}
$$

Labels need not be consistent;

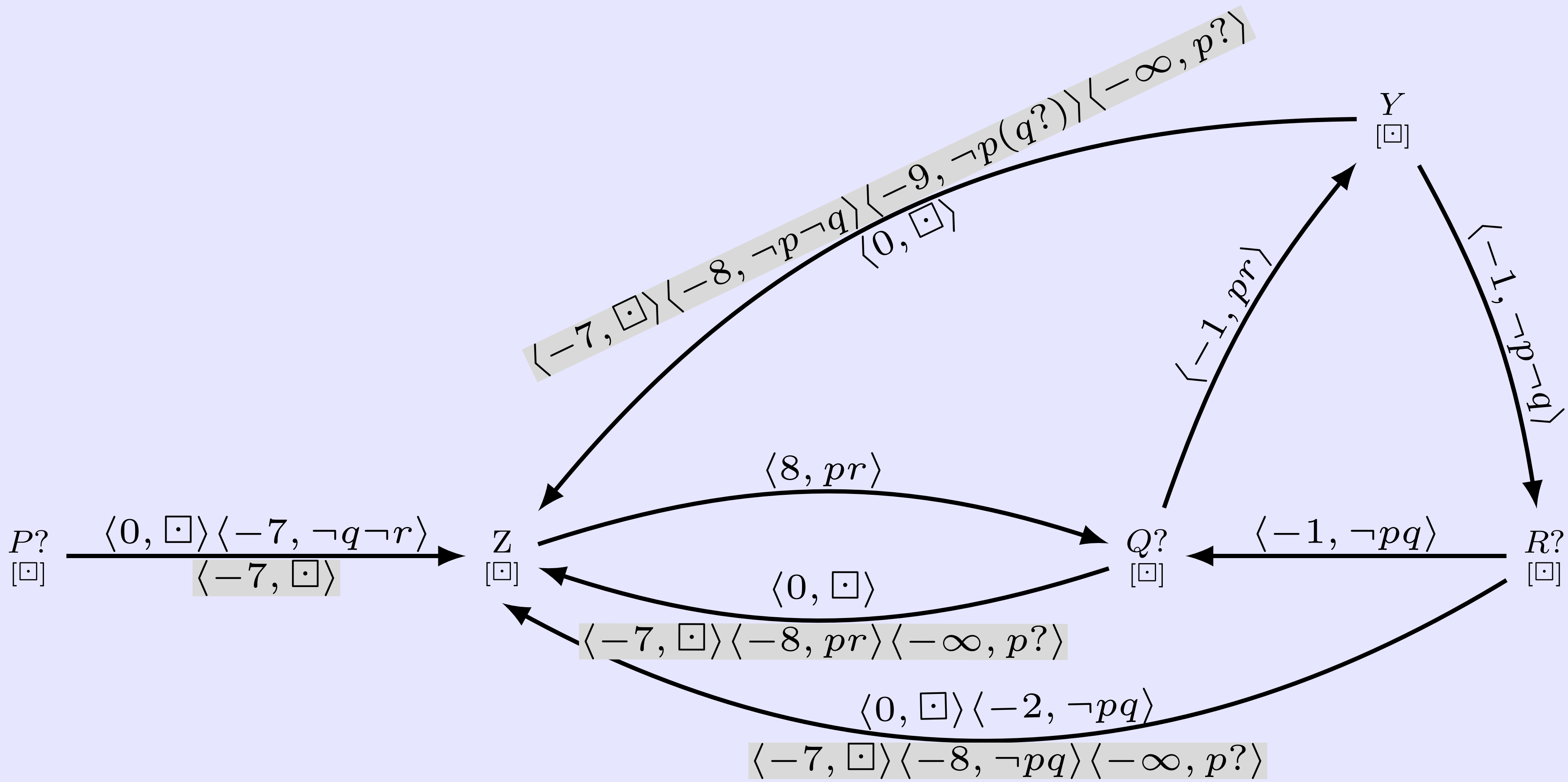
Lefthand weight must be negative;

Generated weight is max of pre-existing weights.

Negative Q-Loop Example

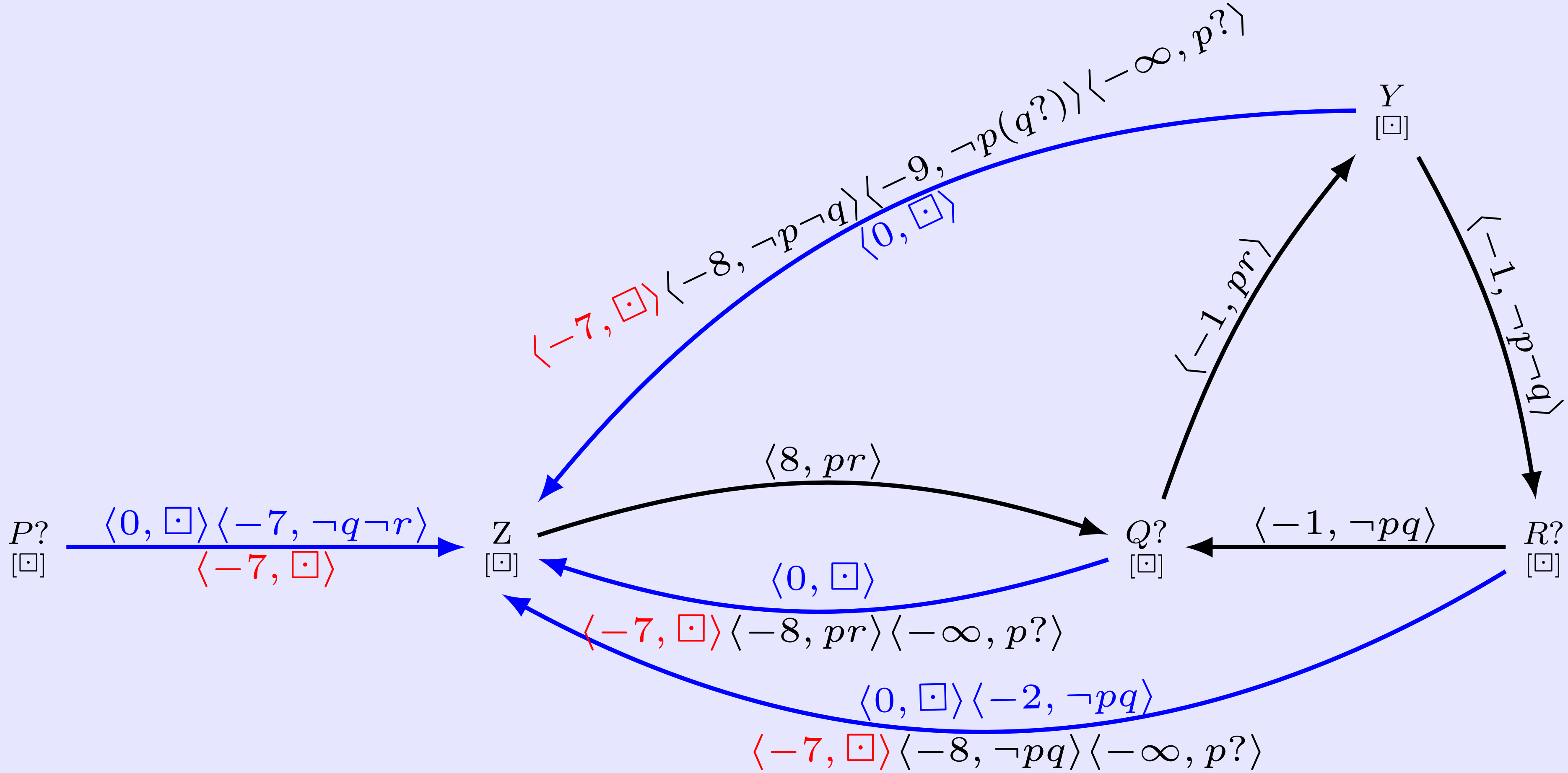


Completing the Propagation



The *Spreading Lemma*

The minimum lower-bound constraint $\langle -7, \boxdot\rangle$ has spread to all unexecuted time-points.



DC-Checking Alg. for CSTNs

- The DC-Checking Alg. does exhaustive propagation
- Returns `NO` if any negative loop with a consistent label is ever found; otherwise returns `YES`.
- In positive cases, constructs *earliest-first* strategy, which is viable due to the spreading lemma.
- Although exponential-time in the worst case, shown to be practical across a variety of sample networks.

(Hunsberger, Posenato, and Combi 2015)

The *Earliest-First* Strategy

- Keep track of *current partial scenario* (CPS), π.
  Initially $\pi = \boxdot$.

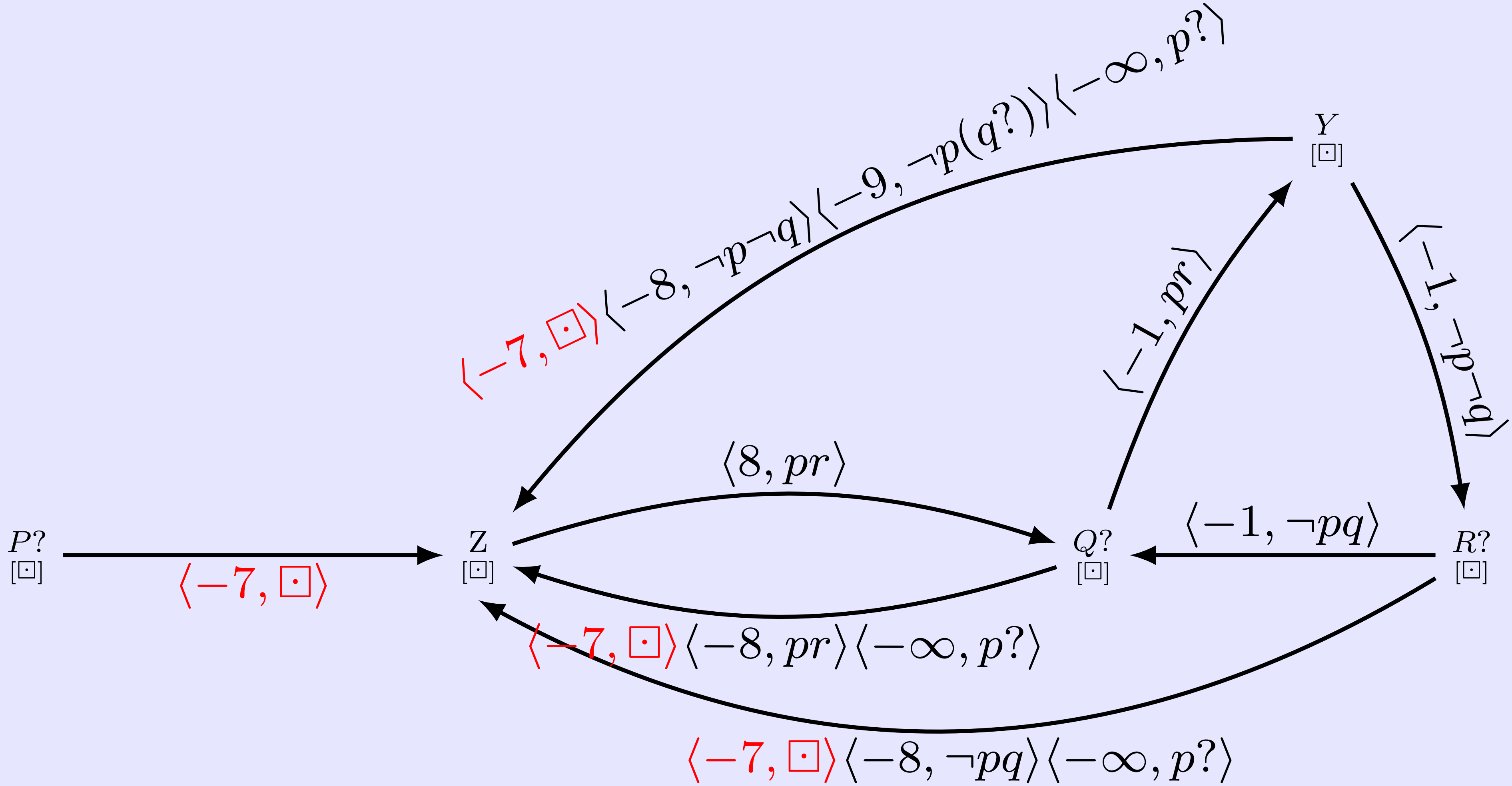
- After each execution event, compute *effective lower bound* (ELB) for each as-yet-unexecuted time-point.

- $ELB(X, \pi)$ restricts attention to lower bounds for X whose labels are applicable to π.

- Execute X next if it has the minimum ELB value.

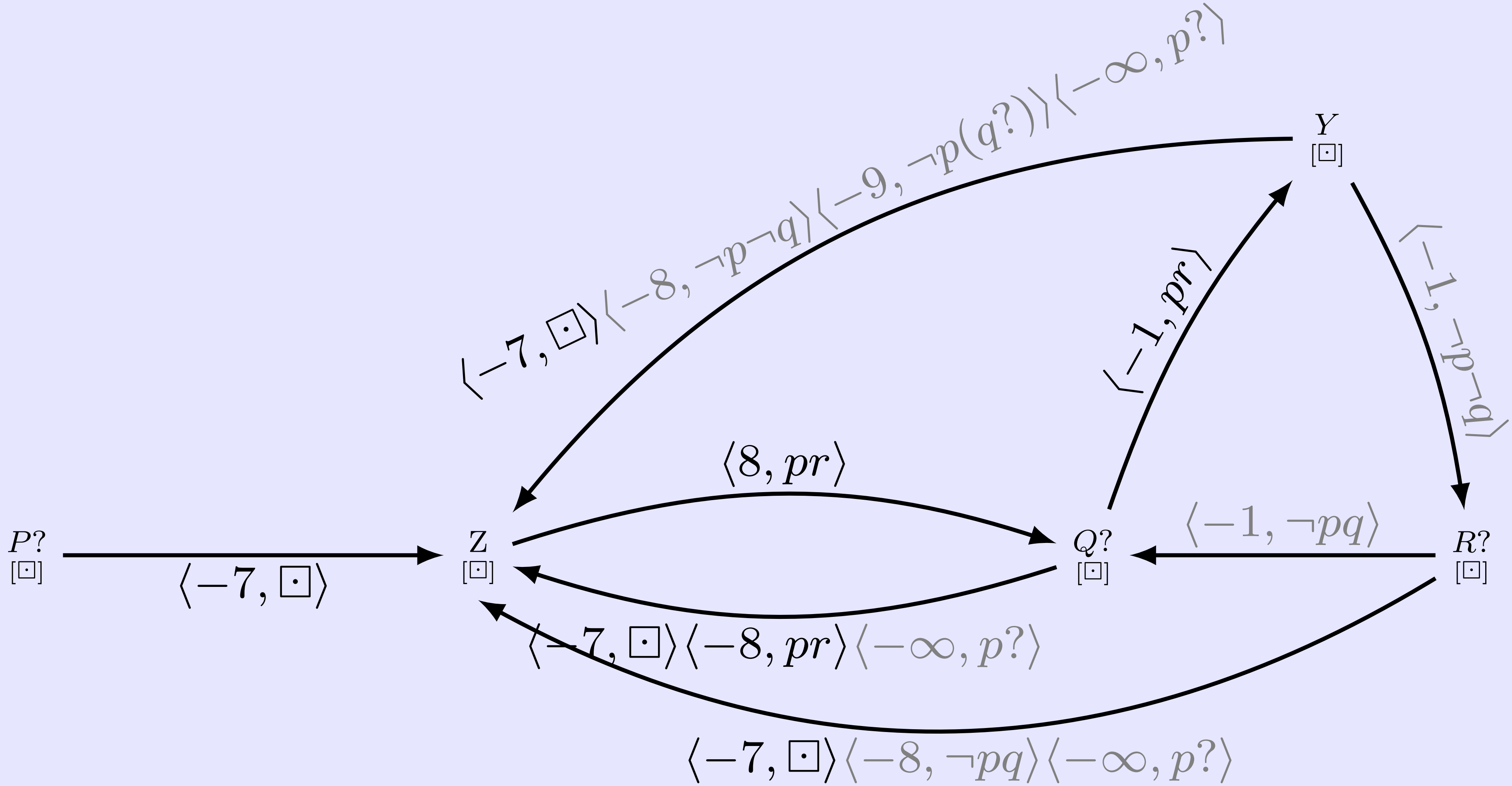
Sample Execution

$\pi = \boxdot, \quad Z = 0, \quad ELB(P?, \boxdot) = -7;\quad$ execute $P? = 7.$



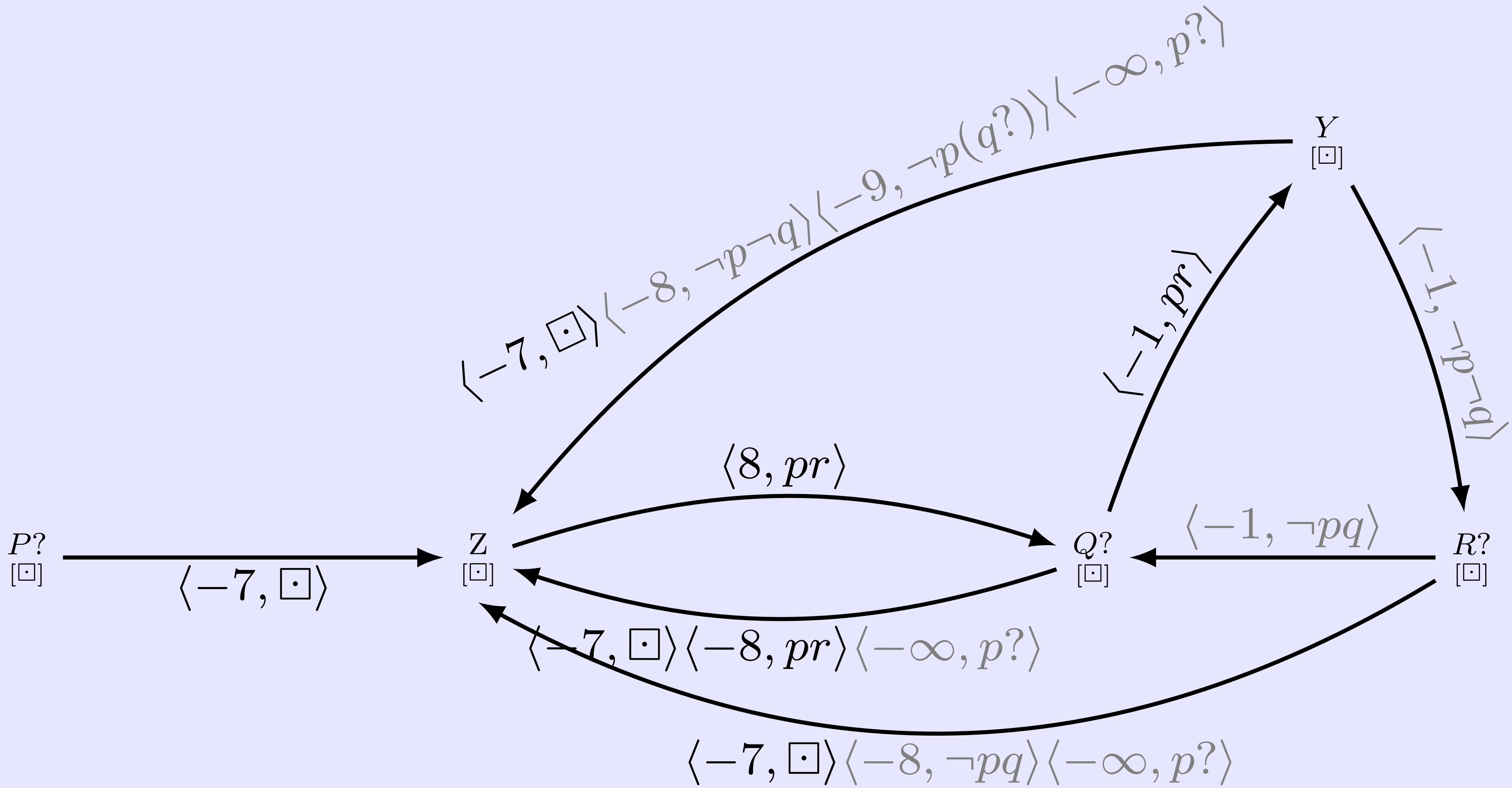
Sample Execution (ctd.)

Suppose $p = true$. $\pi = p$; $ELB(Y, p) = 7 = ELB(R?, p)$.
So execute $Y = 7$ and $R? = 7$.



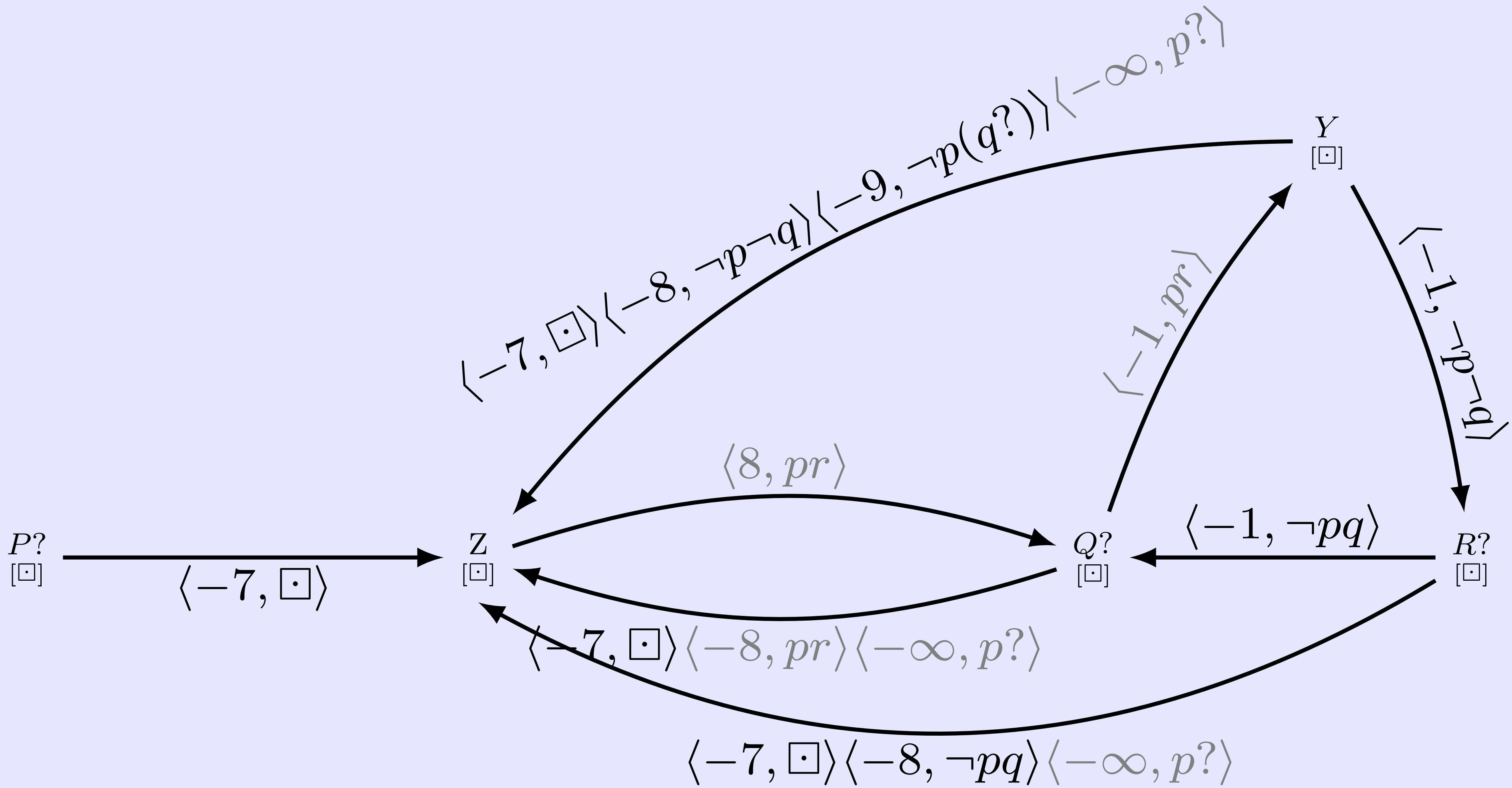
Sample Execution (ctd.)

Suppose $r = true$. $\pi = pr$; $ELB(Q?, p) = 8$.
So execute $Q? = 8$.



Alternative Execution

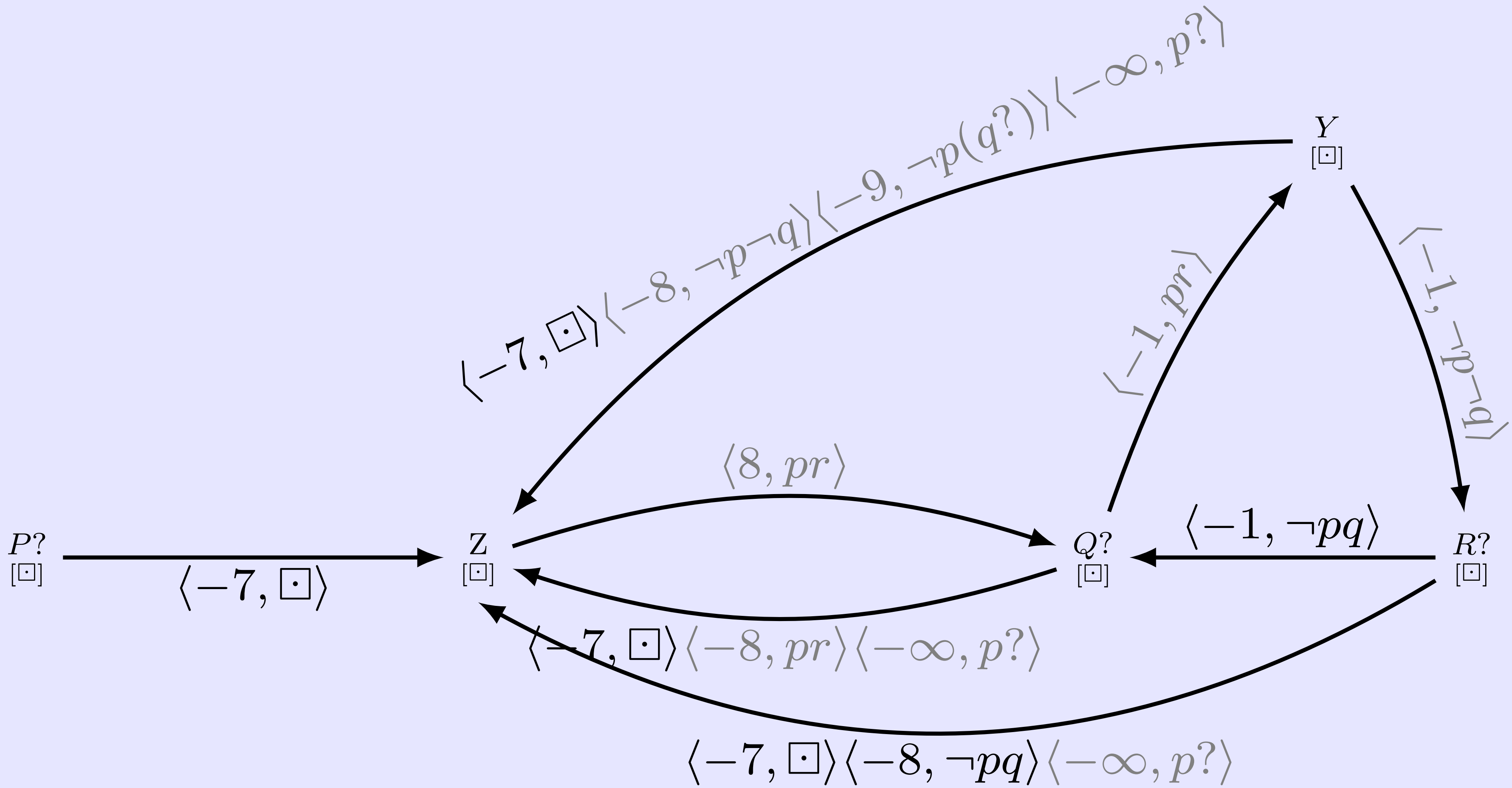
Suppose $p = false$. $\pi = \neg p$; $ELB(Q?, p) = 7$.
So execute $Q? = 7$.



Alternative Execution (ctd.)

Suppose $q = true$. $\pi = \neg pq$; $ELB(Y, \neg pq) = 7$.

So execute $Y = 7$. Afterward, execute $R? = 8$.



Bounded Reaction Time

- ϵ-dynamic controllability requires bounded reaction time $\epsilon > 0$ (Comin and Rizzi 2015).
- Propagation-based ϵ-DC checking algorithm (Hunsberger and Posenato 2016).
- Semantics of instantaneous reactivity for CSTNs (Cairo, Comin, and Rizzi 2016).

CSTN Summary

- Theory of dynamic consistency for CSTNs very solid (instantaneous/non-instantaneous reactivity; bounded reaction time).
- Several competing DC-checking algorithms—all are exponential, but propagation-based algorithm shows promise.
- More work to do on flexible execution.

CSTNUs

CSTNUs

- A *Conditional Simple Temporal Network with Uncertainty* (CSTNU) combines contingent links from STNUs and observation time-points from CSTNs.
- Sound-but-not-complete DC-checking algorithm presented years ago (Combi, Hunsberger, and Posenato 2013).
- Sound-and-complete DC-checking algorithm that extends rules for STNUs and CSTNs is forthcoming!

CDTNUs

Adding Disjunction to CSTNUs

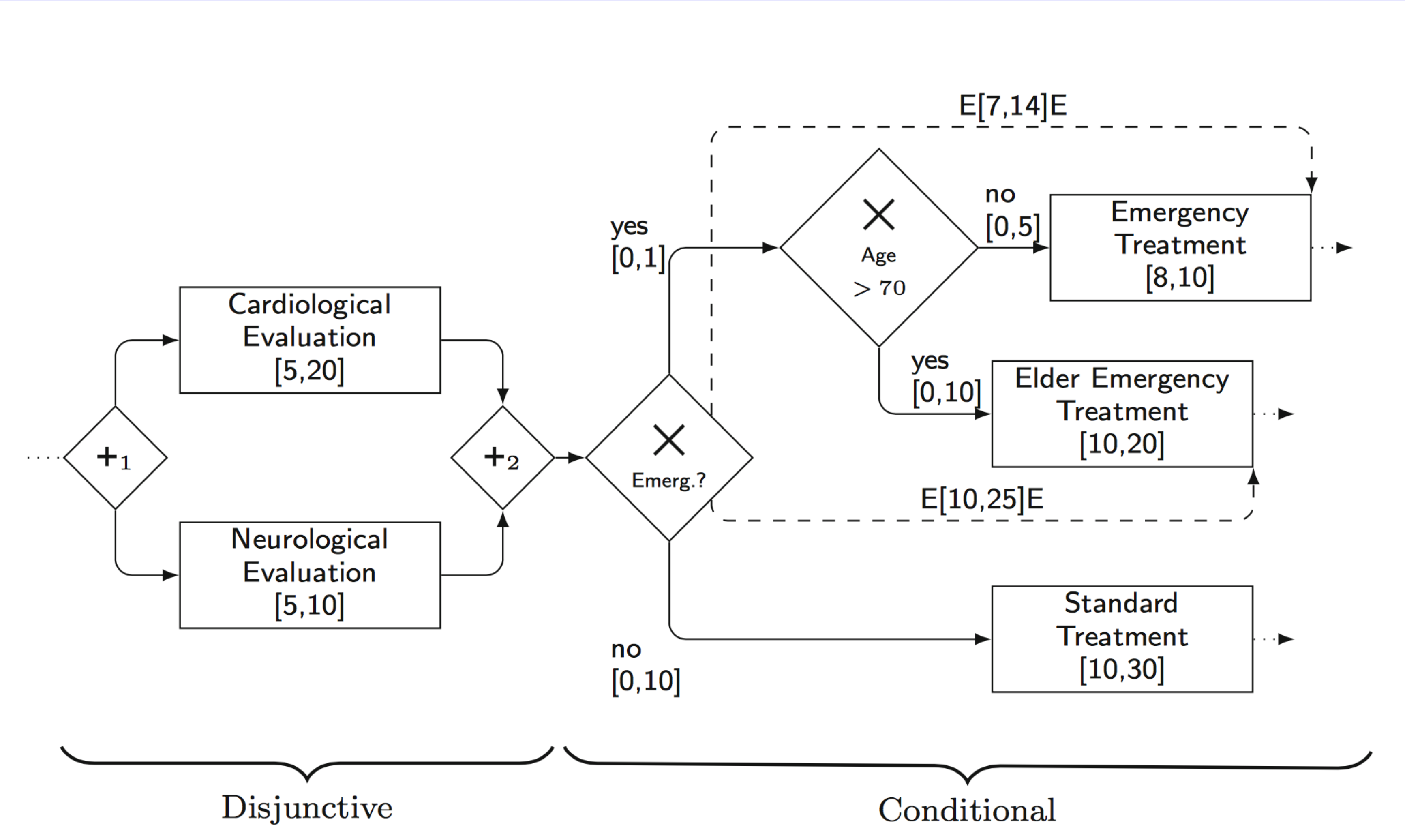
- A *Conditional Disjunctive Temporal Network with Uncertainty* (CDTNU) augments a CSTNU to include disjunctive constraints.

- Possible to convert the DC-checking problem for CDTNUs into a *controller-synthesis* problem for a *Timed Game Automaton* (TGA)*.

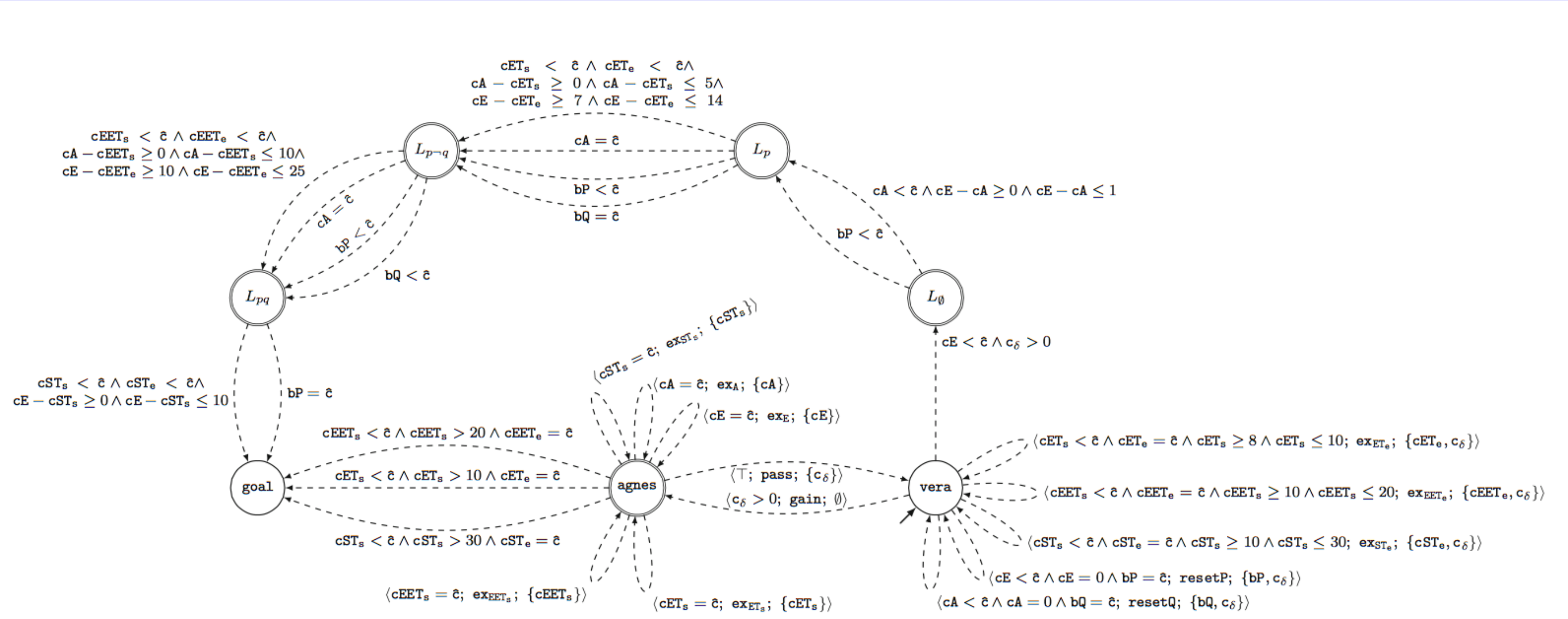
- Highlights connections between temporal networks and TGAs, but algorithm not yet practical.

* (Cimatti et al. 2016)

Sample Workflow



TGA Encoding of Workflow



Conclusions

Conclusions

- Theoretical foundations for a variety of temporal networks are quite solid.
- STNs have been incorporated into planning and scheduling applications for over a decade.
- $O(N^3)$-time DC-checking/dispatchability algorithm for STNUs makes them ready for prime time.
- Propagation-based algorithms for CSTNs and CSTNUs show promise.

References

- Cairo, M.; Comin, C.; and Rizzi, R. 2016. Instantaneous reaction-time in dynamic-consistency checking of conditional simple temporal networks. In *Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning (TIME-2016)*.
- Cesta, A., and Oddi, A. 1996. Gaining efficiency and flexibility in the simple temporal problem. In *Proceedings of the Third International Workshop on Temporal Representation and Reasoning (TIME-96)*, 45–50. IEEE.
- Cimatti, A.; Hunsberger, L.; Micheli, A.; Posenato, R.; and Roveri, M. 2014. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *21st International Symposium on Temporal Representation and Reasoning (TIME-2014), Verona, Italy*, 27–36. IEEE Computer Society.
- Cimatti, A.; Hunsberger, L.; Micheli, A.; Posenato, R.; and Roveri, M. 2016. Dynamic controllability via timed game automata. *Acta Informatica* 53:681–722.
- Combi, C.; Hunsberger, L.; and Posenato, R. 2013. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In Filipe, J., and Fred, A., eds., *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART-2013)*, volume 2, 144–156. SCITEPRESS.
- Comin, C., and Rizzi, R. 2015. Dynamic consistency of conditional simple temporal networks via mean payoff games: a singly-exponential time dc-checking. In Grandi, F.; Lange, M.; and Lomuscio, A., eds., *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, 19–28. IEEE CPS.
- Conrad, P. R., and Williams, B. C. 2011. Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research (JAIR)* 42:607–659.
- Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. Cambridge, MA: The MIT Press.
- Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.
- Demetrescu, C., and Italiano, G. 2002. A new approach to dynamic all pairs shortest paths. Technical Report ALCOMFT-TR-02-92, ALCOM-FT. To appear in Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03), San Diego, California, June 2003.
- Even, S., and Gazit, H. 1985. Updating distances in dynamic graphs. *Methods of Operations Research* 49:371–387.
- Gerevini, A.; Perini, A.; and Ricci, F. 1996. Incremental algorithms for managing temporal constraints. Technical Report IRST-9605-07, IRST.
- Hunsberger, L., and Posenato, R. 2016. Checking the dynamic consistency of conditional simple temporal networks with bounded

reaction times. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS-2016)*.

- Hunsberger, L.; Posenato, R.; and Combi, C. 2015. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal networks. In Grandi, F.; Lange, M.; and Lomuscio, A., eds., *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, 4–18. IEEE.
- Hunsberger, L. 2008. A practical temporal constraint management system for real-time applications. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-2008)*, 553–557. IOS Press.
- Hunsberger, L. 2009. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In *Proceedings of the 16th International Symposium on Temporal Representation and Reasoning (TIME-2009)*, 155–162. IEEE Computer Society.
- Hunsberger, L. 2010. A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In *Proceedings of the 17th International Symposium on Temporal Representation and Reasoning (TIME-2010)*, 121–128. IEEE.
- Hunsberger, L. 2013a. A faster execution algorithm for dynamically controllable STNUs. In Sanchez, C.; Venable, K. B.; and Zimanyi, E., eds., *Proceedings of the 20th International Symposium on Temporal Representation and Reasoning (TIME-2013)*. IEEE Computer Society.
- Hunsberger, L. 2013b. Magic loops in simple temporal networks with uncertainty. In *Proceedings of the Fifth International Conference on Agents and Artificial Intelligence (ICAART-2013)*.
- Hunsberger, L. 2015. Efficient execution of dynamically controllable simple temporal networks with uncertainty. *Acta Informatica* 53(2):89–147.
- Morris, P. H., and Muscettola, N. 2005. Temporal dynamic controllability revisited. In *The Twentieth National Conference on Artificial Intelligence (AAAI-2005)*, 1193–1198. The MIT Press.
- Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 494–499. Morgan Kaufmann.
- Morris, P. 2006. A structural characterization of temporal dynamic controllability. In *Principles and Practice of Constraint Programming (CP 2006)*, volume 4204 of *Lecture Notes in Computer Science*. Springer. 375–389.
- Morris, P. 2014. Dynamic controllability and dispatchability relationships. In H., S., ed., *Integration of AI and OR Techniques in Constraint Programming, CPAIOR 2014*, volume 8451 of *Lecture Notes in Computer Science*. Springer.
- Morris, P. 2016. The mathematics of dispatchability revisited. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS-2016)*.

- Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*.
- Ramalingam, G., and Reps, T. 1996. On the computational complexity of dynamic graph problems. *Theoretical Computer Science* 158:233–277.
- Ramalingam, G.; Song, J.; Joskowicz, L.; and Miller, R. 1999. Solving systems of difference constraints incrementally. *Algorithmica* 23:261–275.
- Rohnert, H. 1985. A dynamization of the all pairs least cost path problem. In Mehlhorn, K., ed., *2nd Symposium of Theoretical Aspects of Computer Science (STACS 85)*, volume 182 of *Lecture Notes in Computer Science*. Springer. 279–286.
- Tsamardinos, I.; Muscettola, N.; and Morris, P. 1998. Fast transformation of temporal plans for efficient execution. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. Cambridge, MA: The MIT Press. 254–261.
- Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints* 8:365–388.