The Dynamic Controllability of Simple Temporal Networks with Uncertainty

Luke Hunsberger
Vassar College
Poughkeepsie, NY, USA

ICAPS-2013 Tutorial
June 10, 2013

Motivating Applications for STNUs

- Agent controlling remote spacecraft
- Fleets of autonomous spacecraft
- Business manufacturing processes
- Medical treatment processes

$\Rightarrow$ *Temporal constraints among actions*
$\Rightarrow$ *Actions with uncertain durations*

Good News

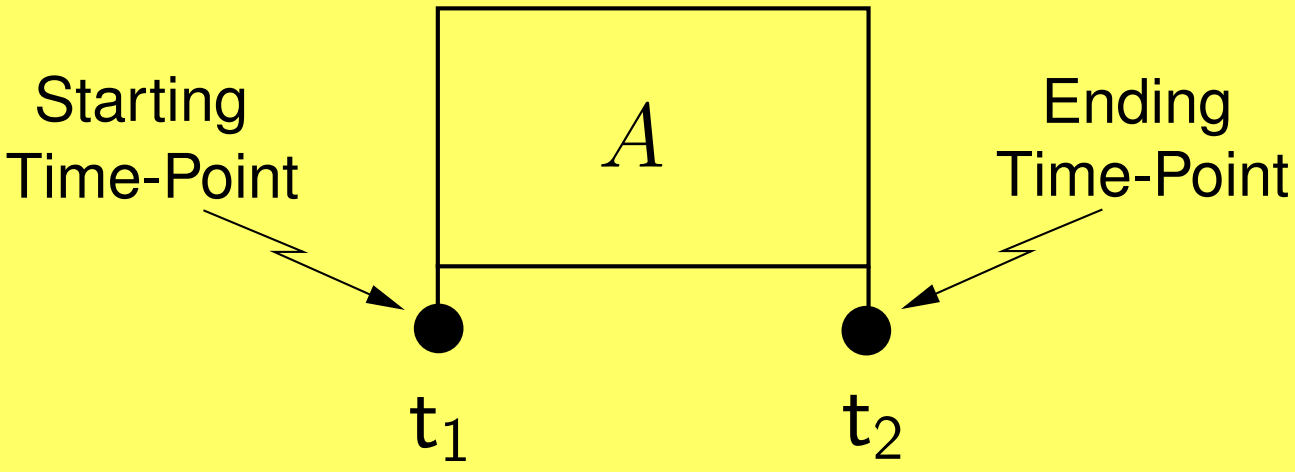
- STNUs represent temporal constraints among actions, including actions with uncertain durations
- Key property: dynamic controllability (DC)
- DC checking can be done in $O(N^4)$ time
- Updates during execution of DC networks requires $O(N^2)$ time per execution event
- Theory of STNUs mature, but still growing

Outline

- Simple Temporal Networks (STNs)
- STNs with Uncertainty (STNUs)
- Dynamic Controllability (DC)
- DC-Checking Algorithms
- Executing STNUs
- Magic Loops in STNUs
- Conclusions

Simple Temporal Networks

Temporal Constraints on an Action



$t_1 \geq 4$ (A starts at or after 4)

$t_2 \leq 12$ (A ends at or before 12)

$3 \leq t_2 - t_1 \leq 6$ (A's duration in $[3, 6]$)

Temporal Constraints on Airline Travel

Goal: Fly from NYC to Rome

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

Simple Temporal Network (STN)*

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

- $\mathcal{T}$ is a set of time-point variables: $\{t_0, t_1, \ldots, t_{n-1}\}$ 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)

Solutions & Consistency

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

  $$\{t_0 = w_0,\ t_1 = w_1,\ \ldots,\ t_{n-1} = w_{n-1}\}$$

  that satisfies all the constraints in $\mathcal{C}$.
- An STN with at least one solution is called *consistent.*

The *Zero* Time-Point Variable

- Frequently, it is useful to fix one of the time-point variables to 0. That "variable" will often be called z.
- Binary constraints involving z are equivalent to unary constraints:

$$t_j - z \leq 5 \iff t_j \leq 5$$

$$z - t_i \leq -3 \iff t_i \geq 3$$

STN for Constrained Action

$$\mathcal{T} = \{z, t_1, t_2\}, \text{ where: } \begin{array}{rcl} z & = & 0 \\ t_1 & = & \text{Start of } A \\ t_2 & = & \text{End of } A \end{array}$$

$$\mathcal{C} = \begin{pmatrix} t_2 - t_1 \leq 6 & (\text{Dur. less than 6}) \\ t_1 - t_2 \leq -3 & (\text{Dur. greater than 3}) \\ z - t_1 \leq -4 & (\text{A starts after 4}) \\ t_2 - z \leq 12 & (\text{A ends before 12}) \end{pmatrix}$$

STN for Constrained Air Travel

$\mathcal{T} = \{z, t_1, t_2, t_3, t_4\}$, $z =$ Noon, June 8.

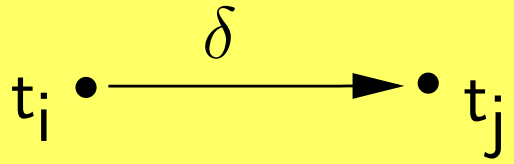
$\mathcal{C} =$

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

Graphical Representation of an STN*

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

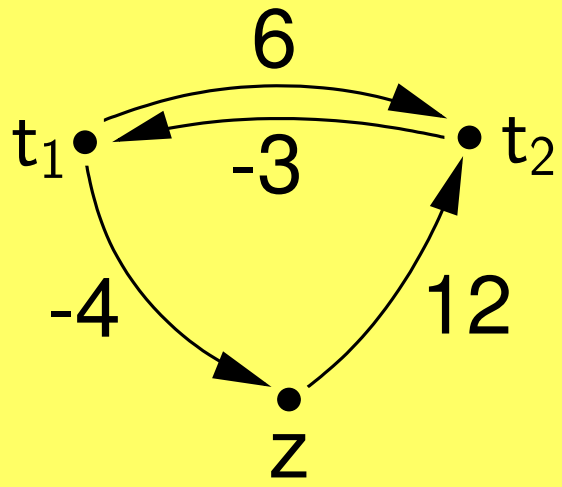
- Time-points in $\mathcal{S}$ $\iff$ nodes in $\mathcal{G}$
- Constraints in $\mathcal{C}$ $\iff$ edges in $\mathcal{E}$:

$t_j - t_i \leq \delta$  

* (Dechter, Meiri, and Pearl 1991)

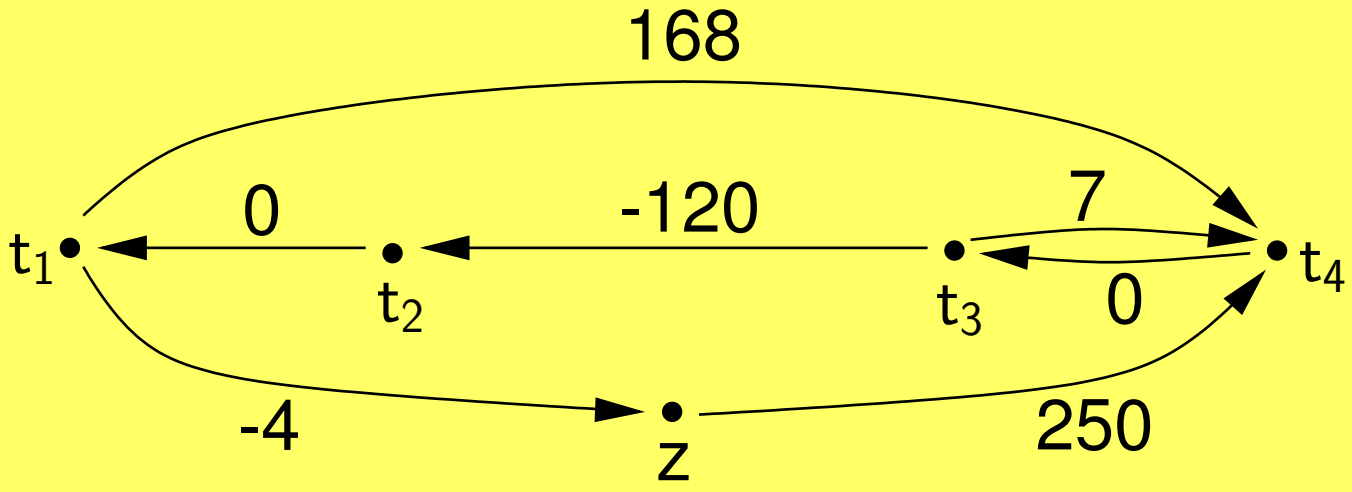
Graph for Action Scenario

$$\mathcal{T} = \{z, t_1, t_2\} \qquad \mathcal{C} = \left\{ \begin{array}{rcr} t_2 - t_1 & \leq & 6 \\ t_1 - t_2 & \leq & -3 \\ z - t_1 & \leq & -4 \\ t_2 - z & \leq & 12 \end{array} \right\}$$



Graph for Airline Scenario

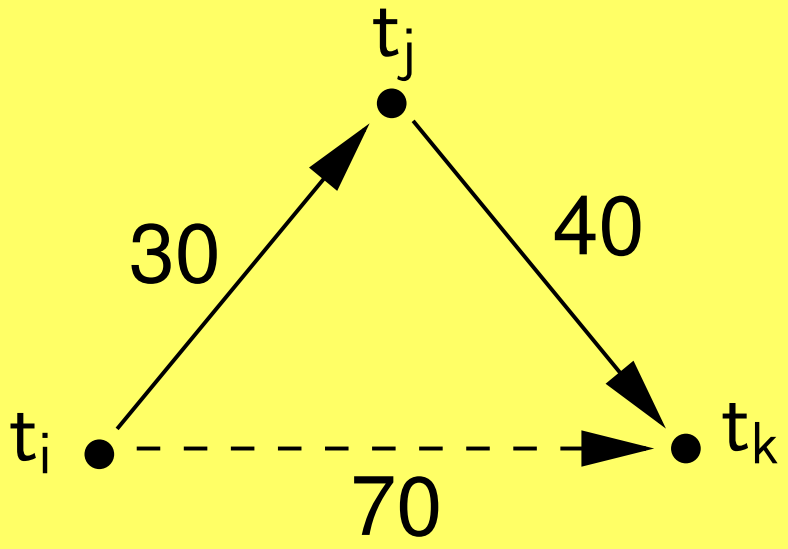
$$\left\{\begin{array}{rcrrcr} z - t_1 & \leq & -4, & t_4 - z & \leq & 250 \\ t_4 - t_1 & \leq & 168, & t_2 - t_3 & \leq & -120 \\ t_4 - t_3 & \leq & 7, & t_1 - t_2 & \leq & 0 \\ t_3 - t_4 & \leq & 0 & & & \end{array}\right\}$$



Implicit Constraints

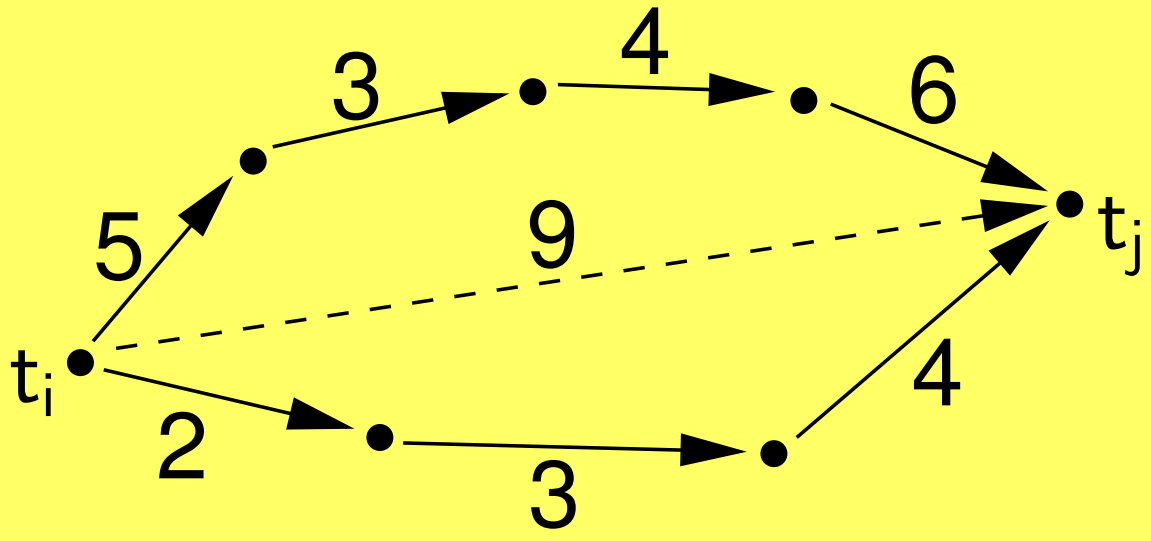
Explicit constraints in $\mathcal{C}$ can combine to form implicit constraints:

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



Chains of Constraints as Paths

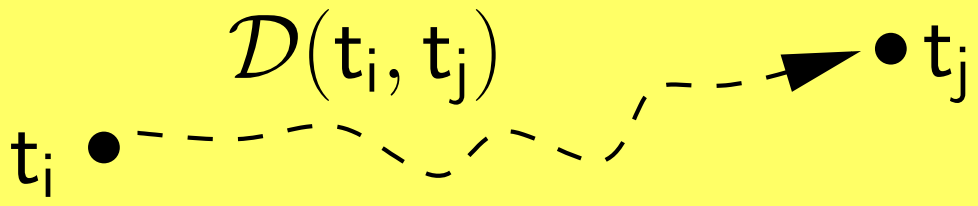
- Chains of constraints in an STN correspond to *paths* in its graph.
- *Stronger/strongest* constraints correspond to *shorter/shortest* paths.



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 $\mathcal{S}$



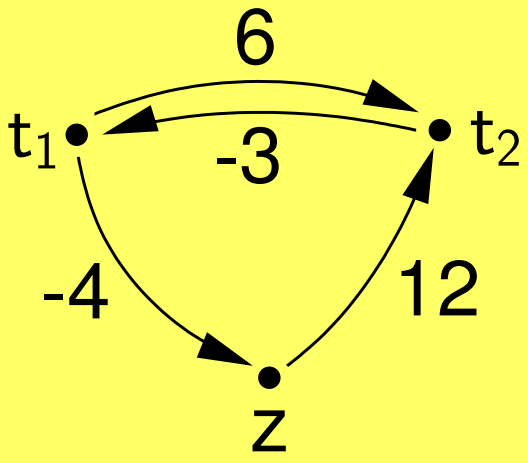
(Dechter, Meiri, and Pearl 1991)

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* Matrix for the STN's graph.*

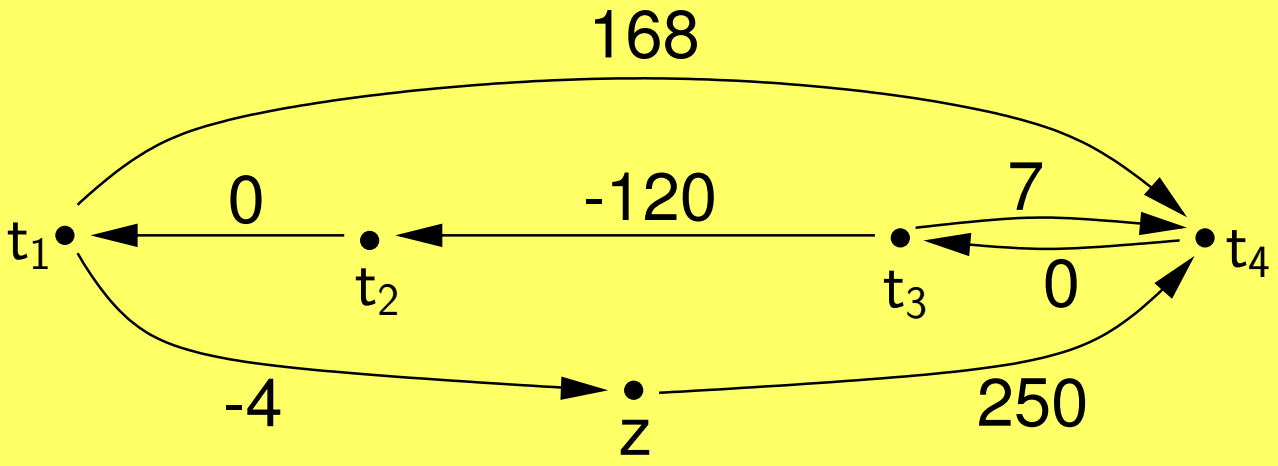
* (Cormen, Leiserson, and Rivest 1990)

Distance Matrix for Action Scenario



$\mathcal{D}$	z	t_1	t_2
z	0	9	12
t_1	-4	0	6
t_2	-7	-3	0

Distance Matrix for Airline Scenario



$\mathcal{D}$	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 for STNs

Given an STN $\mathcal{S}$, with Graph $\mathcal{G}$, and Distance Matrix $\mathcal{D}$, the following are equivalent (Dechter, Meiri, and Pearl 1991):

- $\mathcal{S}$ is consistent.
- Each loop in $\mathcal{G}$ has non-negative length.
- The entries along the main diagonal of $\mathcal{D}$ are non-negative.

Computing $\mathcal{D}$ from Scratch

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

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

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

Executing an STN in Real Time

Any consistent STN can be successfully executed in real time.*

- Time window for X: $[-\mathcal{D}(X, Z), \mathcal{D}(Z, X)]$
- To execute a time-point X at time t:
  Insert constraints: $t \leq X - Z \leq t$
- Update entries involving Z in linear time per execution event—$O(n^2)$ overall.†

* (Dechter, Meiri, and Pearl 1991), † (Hunsberger 2008)

Executing an STN in Real Time (ctd.)

- Transform STN into *dispatchable* form in $O(n^3)$ or $O(n^2 \log n + nm)$ time.
- During execution, only need to propagate bounds to *neighboring* time-points

(Muscettola, Morris, and Tsamardinos 1998)

(Tsamardinos, Muscettola, and Morris 1998)

STNs with Uncertainty

Simple Temporal Network w/ Uncertainty

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

- $\mathcal{T}$ — Time Points: $A, B, C, \ldots, X, Y, \ldots$
- $\mathcal{C}$ — Temporal Constraints: $Y - X \leq \delta$
- $\mathcal{L}$ — 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*

STNU Graph

- Time Points $\iff$ Nodes
- Temporal Constraints $\iff$ Edges

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

- Contingent Links $\iff$ Labeled Edges

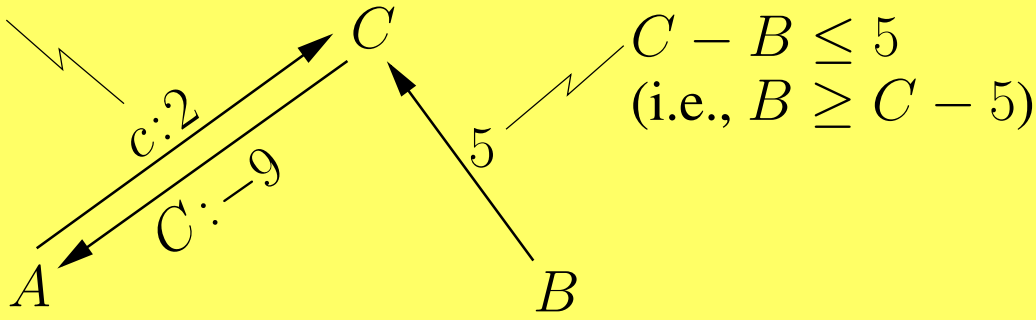
$$C - A \in [\ell, u] \iff A \xrightarrow{c:\ell} C$$

$$A \xleftarrow{C:-u} C$$

STNU Example

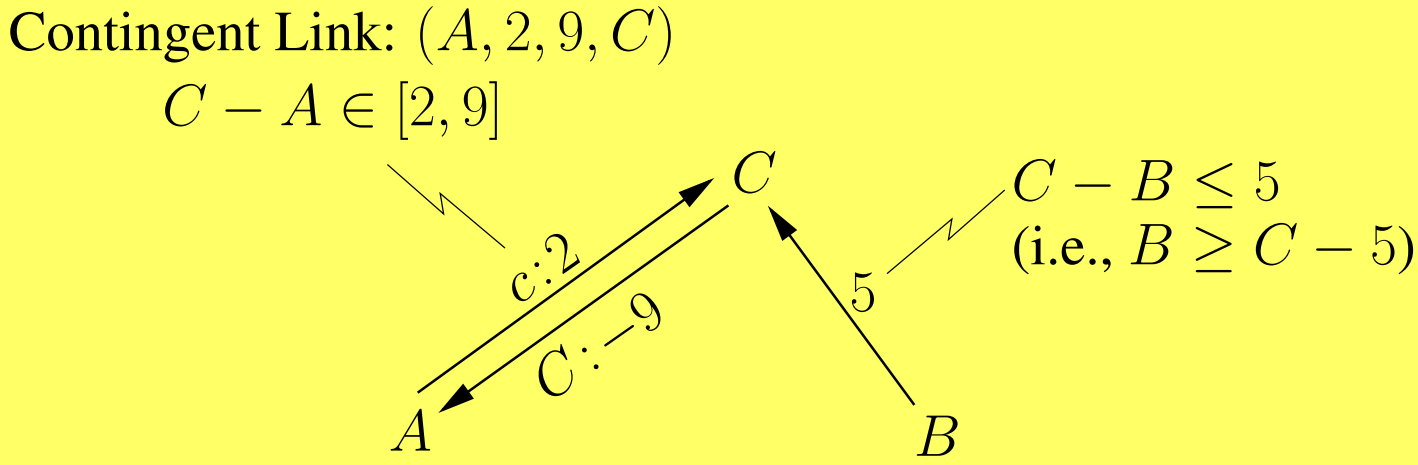
Contingent Link: $(A, 2, 9, C)$

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



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

STNU Example

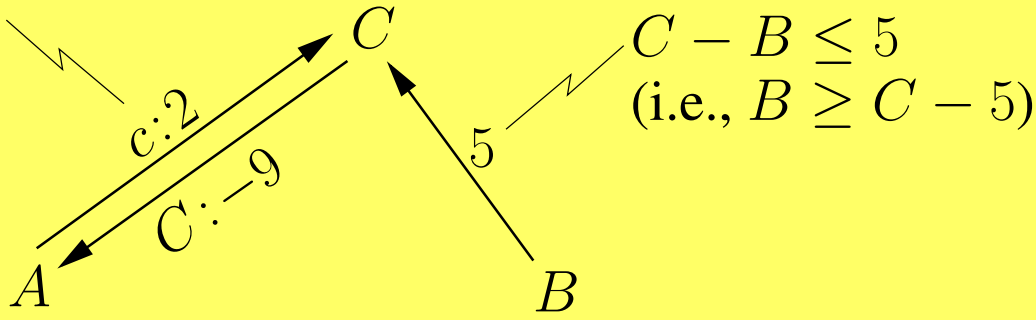


If $A = 0$ and $B = 2$, then problem if $C > 7$.

STNU Example

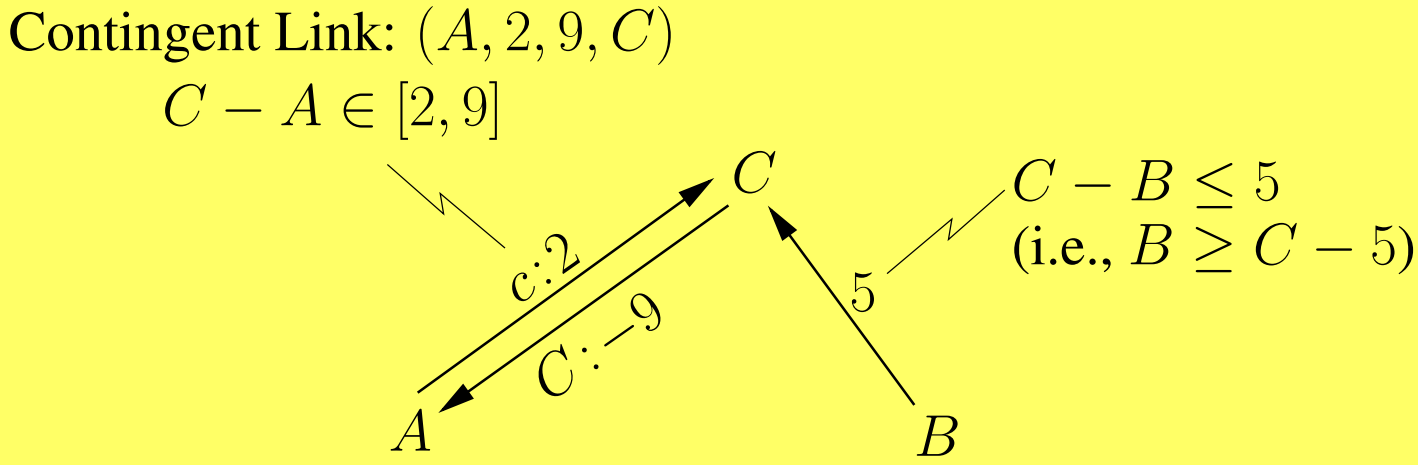
Contingent Link: $(A, 2, 9, C)$

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



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

STNU Example

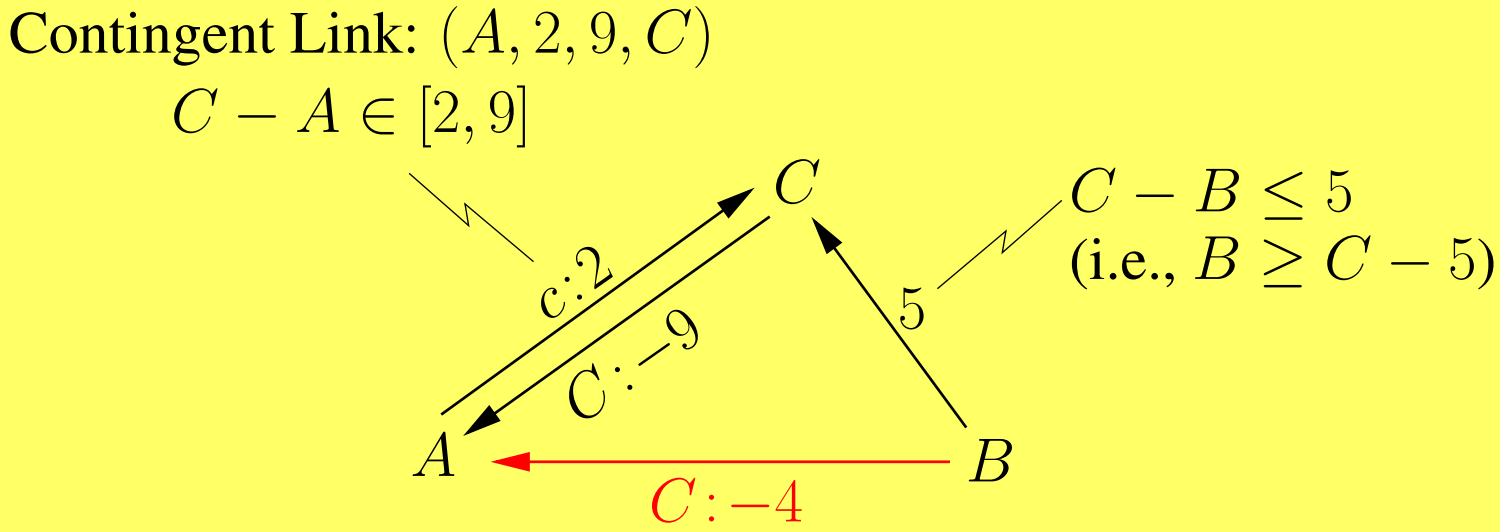


If $A = 0$ and $C = 3$, then $B > 3$ no problem!

Dynamic Controllability (DC)

An STNU is *dynamically controllable* if there exists a *strategy* for executing the *non-contingent* time-points such that *all* of the constraints in the network will be satisfied—*no matter how the durations of the contingent links turn out.*

STNU Example



Strategy: As long as C unexecuted,
B must wait until time 4.

Fundamental Theorem for STNUs

Given an STNU $\mathcal{S}$, with graph $\mathcal{G}$, and SR-distance matrix $\mathcal{D}^*$, the following are equivalent:

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

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

STNU Algorithms

For an STNU with N time-points and K contingent links:

- Checking whether it is DC can be done in $O(N^4)$ time†
- Executing a DC STNU can be done in $O(N^4)$ time*
- $O(N^3)$-time execution alg. forthcoming!

†(Morris 2006); *(Hunsberger 2010)

Dynamic Controllability (DC)

Key Ideas

- For a contingent link, (A, ℓ, u, C), once A is executed, the execution of C is uncontrollable, but guaranteed to be such that $C - A \in [\ell, u]$.
- Strategy for executing time-points can only depend on *past* observations
- Original MMV[†] semantics mixes these together . . .

[†] (Morris, Muscettola, and Vidal 2001)

MMV Semantics: Overview

- Schedule: Assigns times to time-points
- Situation: One way the durations of the contingent links could play out
- Strategy: Mapping from situations to schedules

$\Rightarrow$ Execution happens incrementally, based on partial view of “real” situation

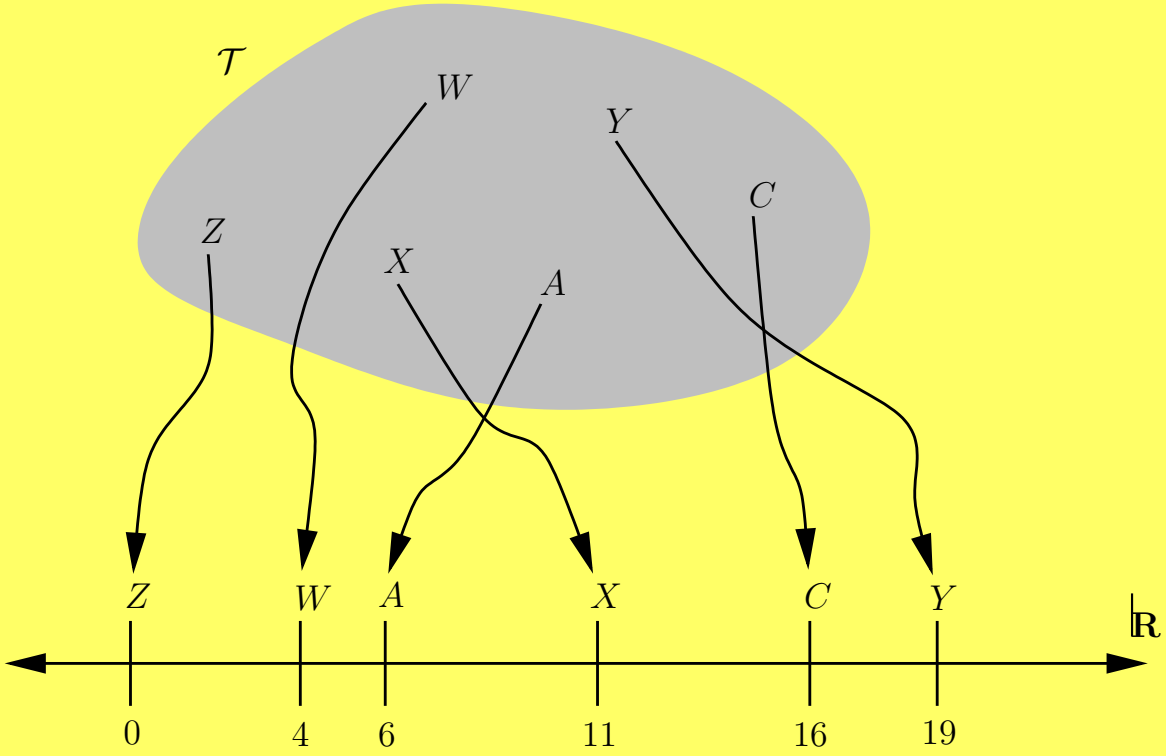
MMV Semantics: Schedules

Given an STNU $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$:

- $\psi : \mathcal{T} \rightarrow \mathbb{R}$ is a (complete) *schedule*
- ψ_X = time of X in schedule ψ
- For any $k \in \mathbb{R}$, $[\psi]^{<k}$ denotes the *pre-history* of ψ relative to time k.[†] (Contains timing info about contingent links that finished before time k in ψ)
- Ψ — the set of all schedules for $\mathcal{S}$

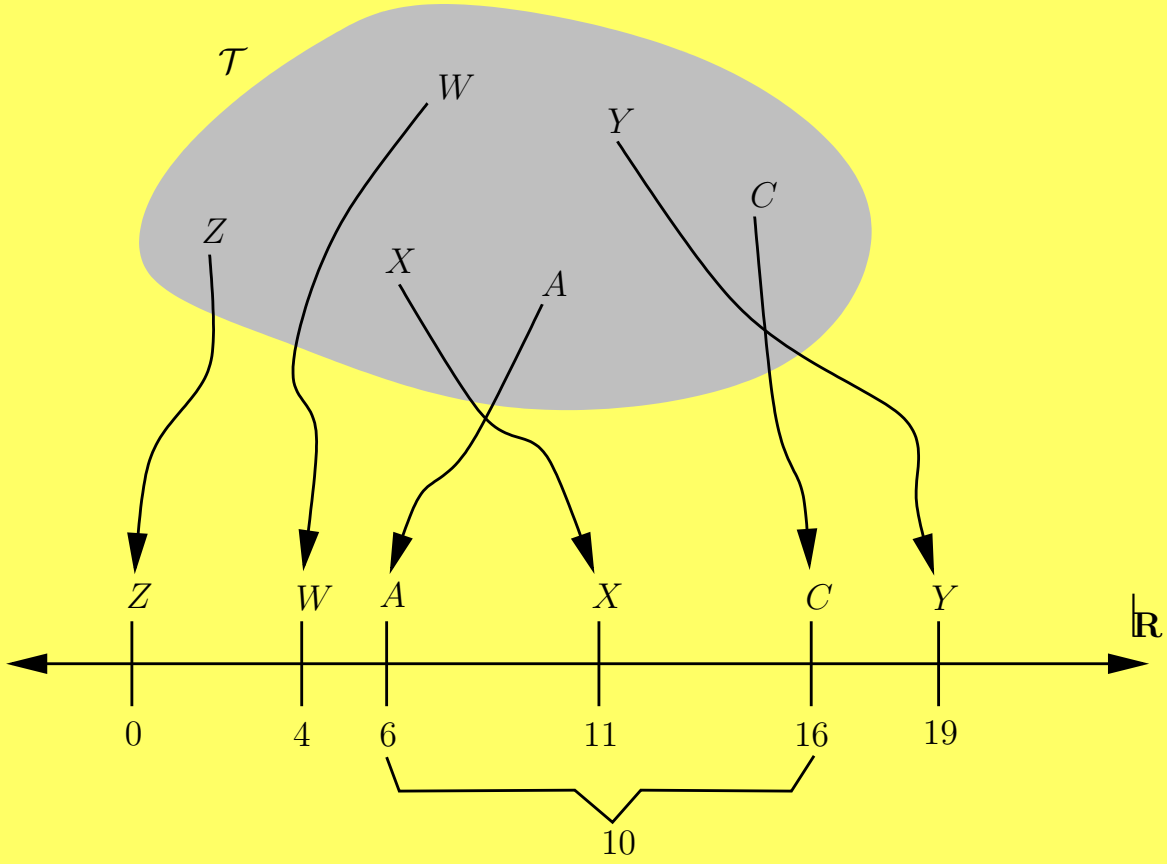
[†] (Hunsberger 2009)

MMV Semantics: Schedules



$$\psi_Z = 0, \quad \psi_W = 4, \quad \psi_A = 6, \quad \ldots$$

MMV Semantics: Schedules



$$[\psi]^{<5} = \emptyset, \quad [\psi]^{<12} = \emptyset, \quad [\psi]^{<19} = \{(C - A = 10)\}$$

MMV Semantics: Situations

For an STNU with K contingent links:

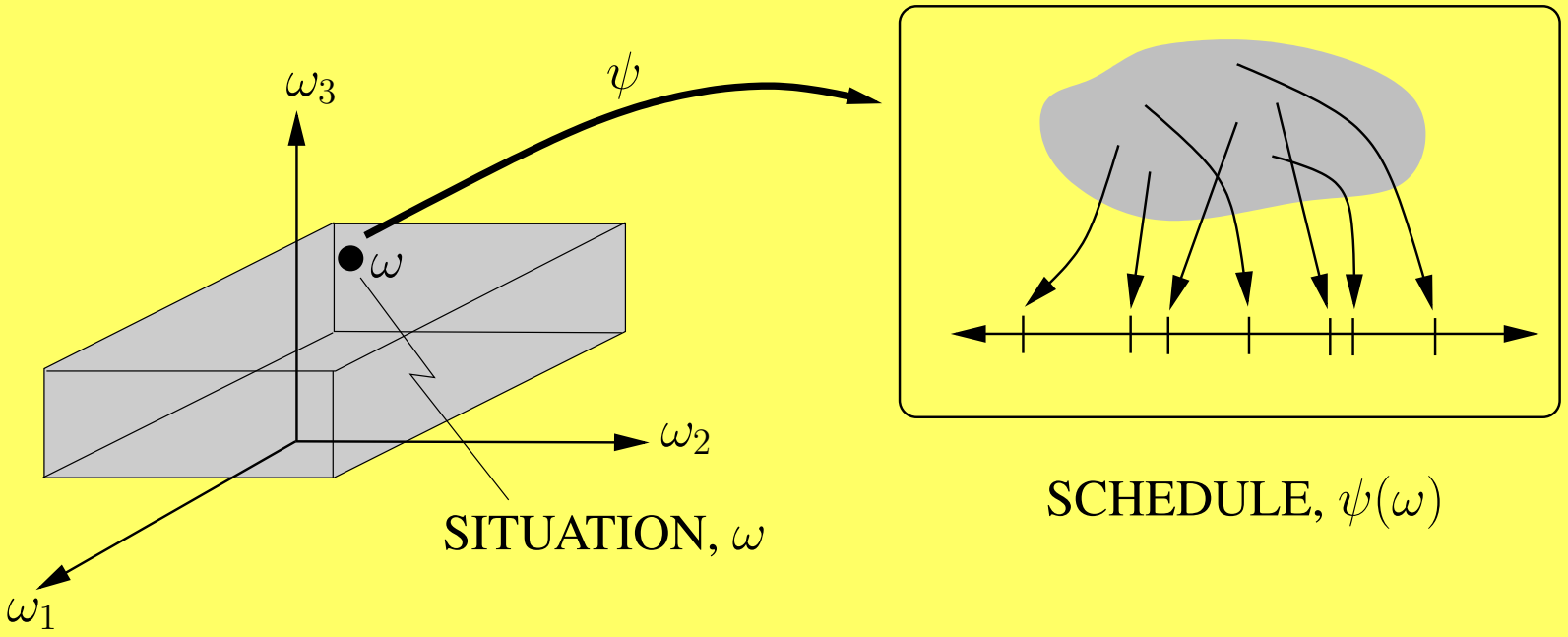
- $\omega = (d_1, d_2, \ldots, d_K)$ is a *situation*, where $\ell_i \leq d_i \leq u_i$, for each i.
- $\Omega = [\ell_1, u_1] \times [\ell_2, u_2] \times \ldots \times [\ell_K, u_K]$ is the space of situations
- Projection: $\mathcal{S}_\omega$ is the *STN* that results from forcing the contingent links to take on the values in ω.

MMV Semantics: Execution Strategies

- A *strategy* is a mapping, $\sigma : \Omega \rightarrow \Psi$, from situations to schedules
- Given a (complete) situation ω,
  $\sigma(\omega)$ is a (complete) schedule.

$\Rightarrow$ But agent builds schedule incrementally, based on partial view of situation!

MMV Semantics: Execution Strategies



MMV Semantics: Valid Strategies

- A strategy σ is *valid* if for each situation ω, the schedule $\sigma(\omega)$ is consistent with the projection $\mathcal{S}_\omega$.

$\Rightarrow$ In other words, a valid strategy does not control the durations of contingent links.

MMV Sem.: Dynamic Execution Strategies

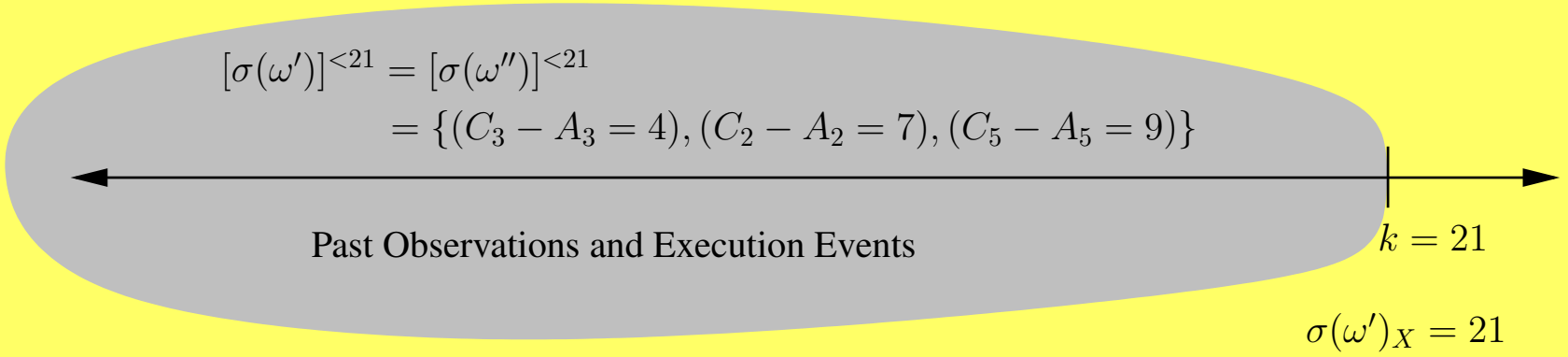
An execution strategy σ is *dynamic* if:

for any situations ω' and ω'',

and any non-contingent time-point $X \in \mathcal{T}$,

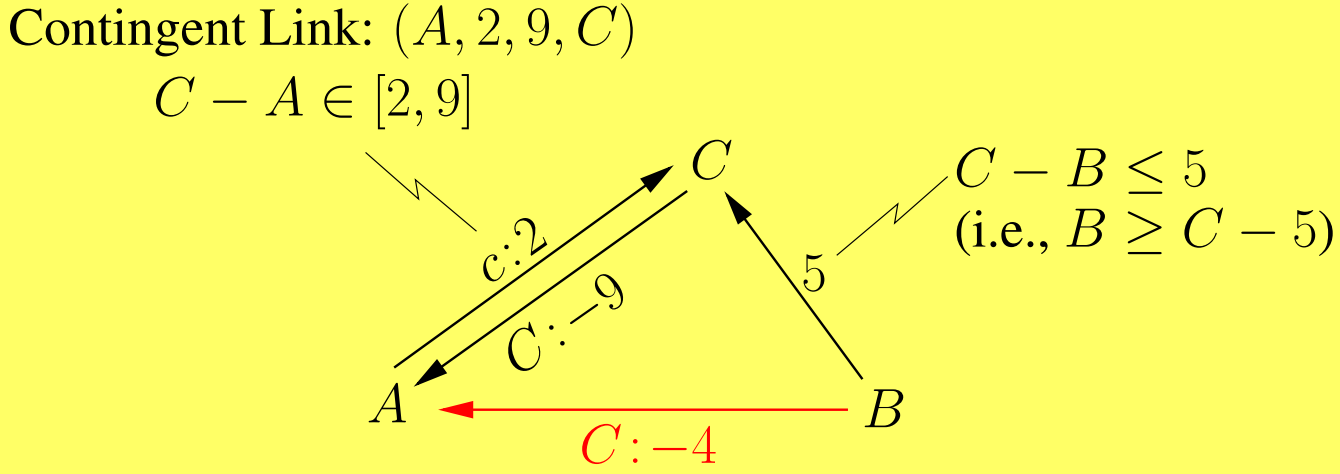
(let $k = [\sigma(\omega')]_X$)

if $[\sigma(\omega')]^{<k} = [\sigma(\omega'')]^{<k}$, then $[\sigma(\omega'')]_X = k$.



MMV Semantics: DC—at last!

An STNU is *dynamically controllable* if there exists an execution schedule for $\mathcal{S}$ that is both valid and dynamic.



Strategy: If $C < 4$, execute B at $C + 1$
otherwise, execute B at 4.

Alternative Semantics

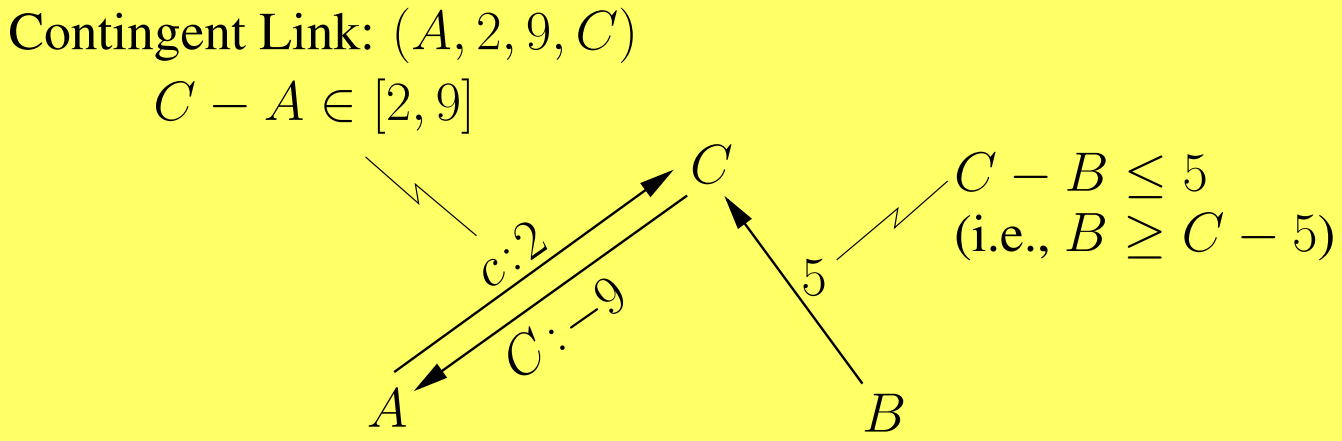
- Real-time execution decisions (RTEDs) based on partial situations
- Makes explicit the decision problem faced during execution
- Outcomes of RTEDs reflect all possible situations
- Equivalent to MMV semantics

(Hunsberger 2009)

Real-Time Execution Decisions

- WAIT:
  Wait for some (already 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.

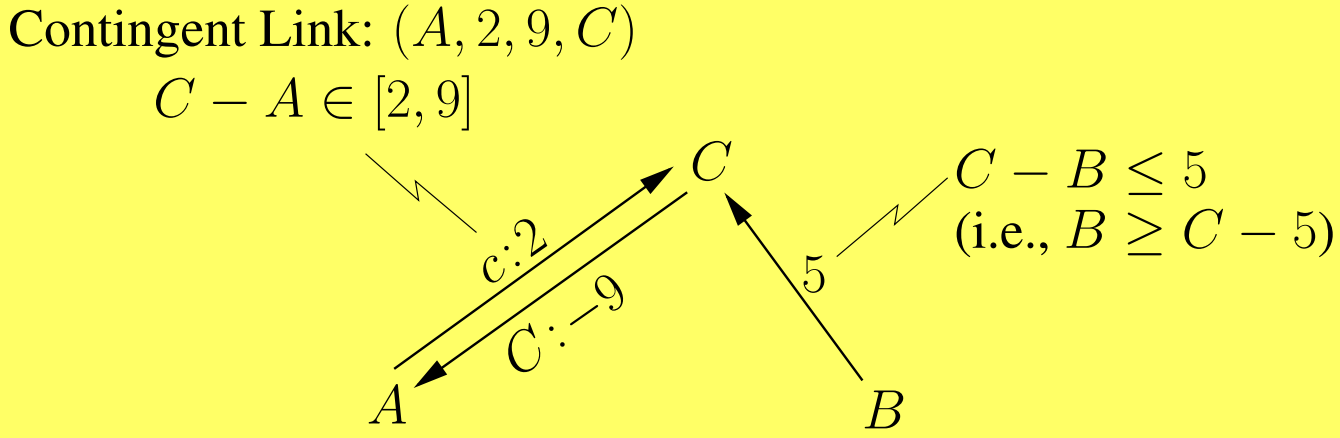
Alternative Semantics: Example



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

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

Alternative Semantics: Example (ctd.)

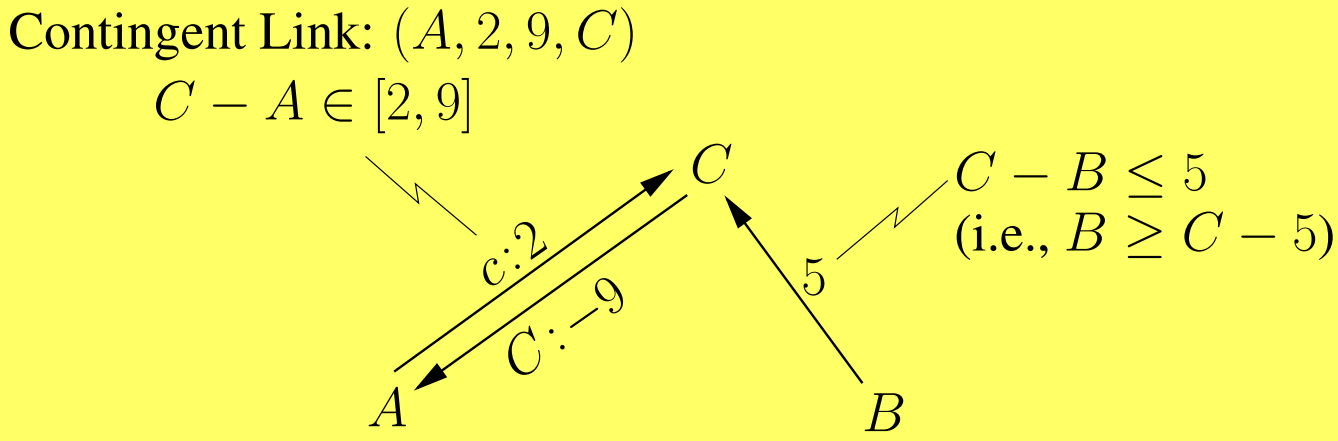


Possible Outcome: C executes at time 2.

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

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

Alternative Semantics: Example (ctd.)



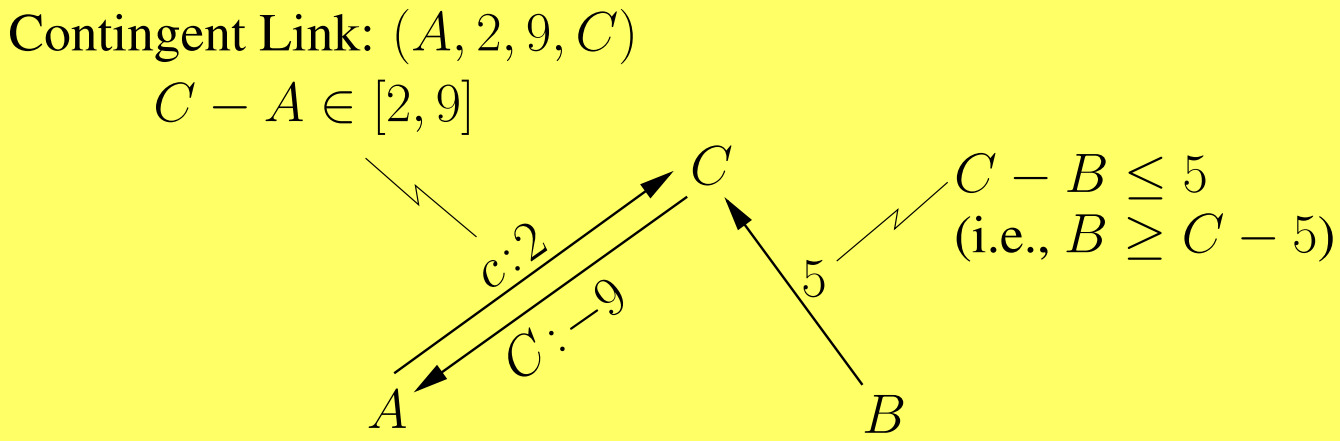
Only Possible Outcome: Execute B at 3.

All done: $A = 0$, $C = 2$, $B = 3$.

(Success!)

Alternative Semantics: Example

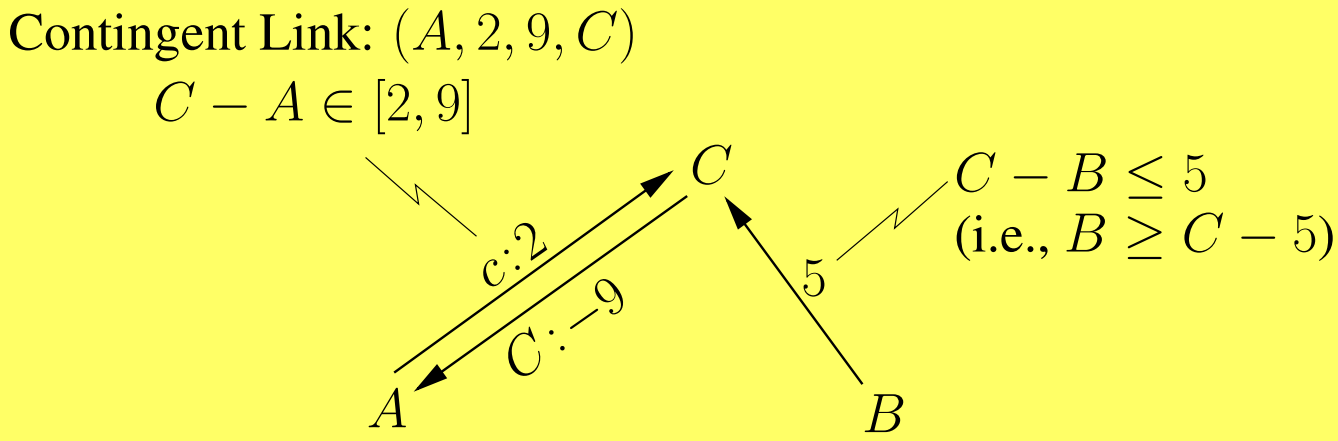
Starting over . . .



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

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

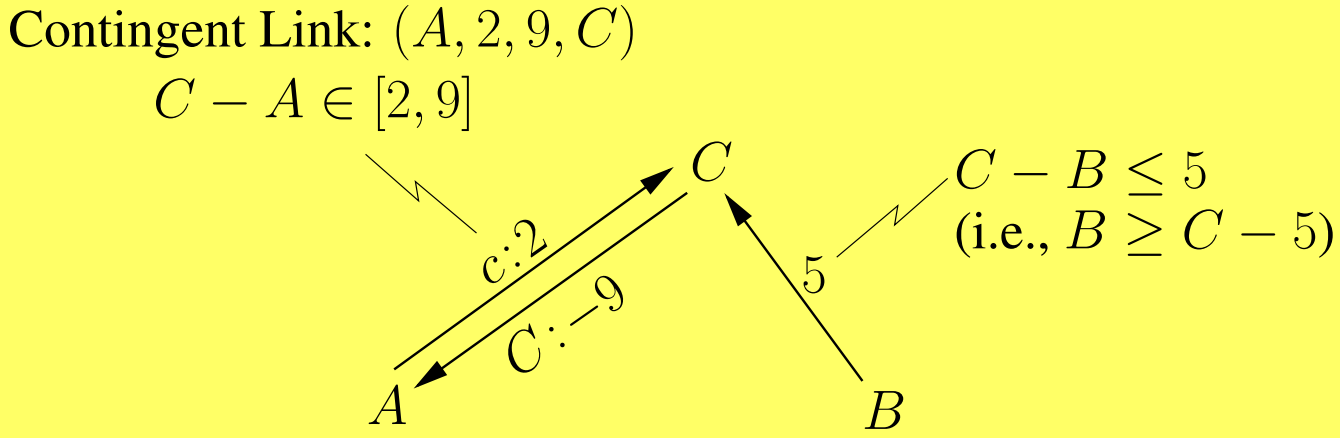
Alternative Semantics: Example (ctd.)



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

Next decision: WAIT (for C to execute)

Alternative Semantics: Example (ctd.)



Possible Outcome: C executes at time 8.

All done: $A = 0$, $B = 4$, $C = 8$.

(Success!)

Two Formulations Equivalent

- There is a one-to-one correspondence between dynamic execution strategies (Morris, Muscettola, and Vidal 2001) and strategies based on real-time execution decisions (Hunsberger 2009)
- RTED-based strategies more practical for agents executing an STNU in real time

DC-Checking Algorithms

History of DC-Checking Algorithms

- MMV-01: *Pseudo-polynomial*

  (Morris, Muscettola, and Vidal 2001)

- MM-05: $O(N^5)$-time

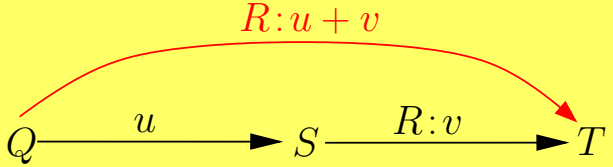
  (Morris and Muscettola 2005)

- Morris-06: $O(N^4)$-time

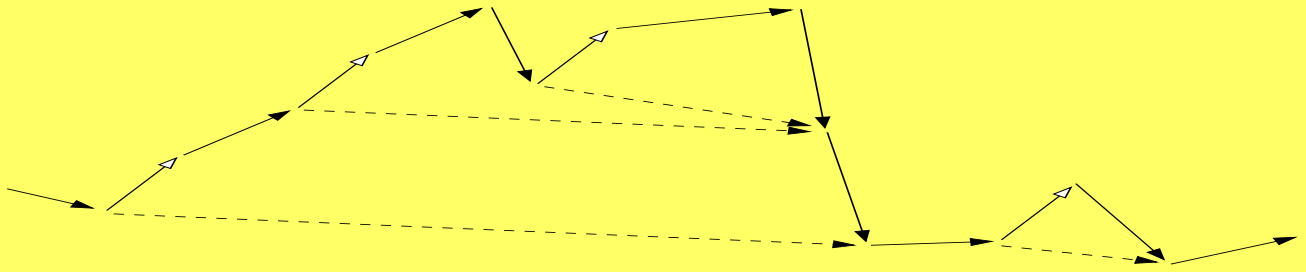
  (Morris 2006)

DC-Checking Overview

- Edge-gen'n/path-transform'n rules†



- Semi-reducible paths*



- DC iff no semi-reducible negative loops*

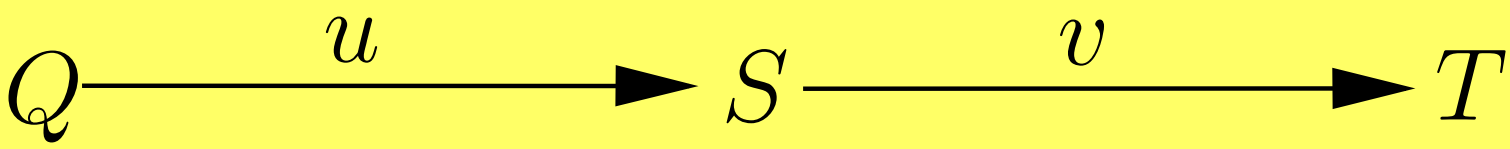
†(Morris and Muscettola 2005); *(Morris 2006)

Edge-Gen'n/Path-Transform'n Rules

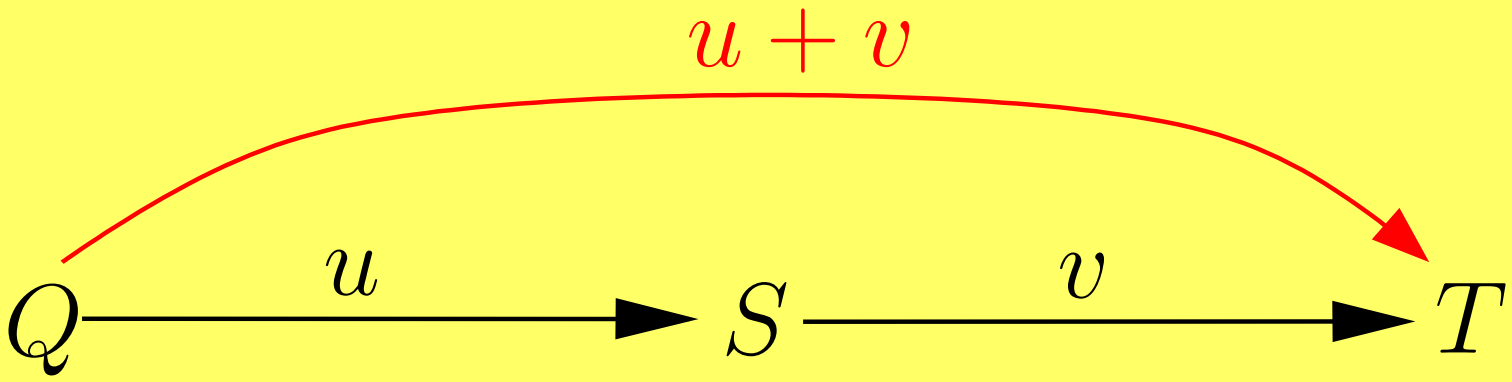
- *No Case* Rule
- *Upper-Case* Rule
- *Lower-Case* Rule
- *Cross-Case* Rule
- *Label-Removal* Rule

(Morris and Muscettola 2005)

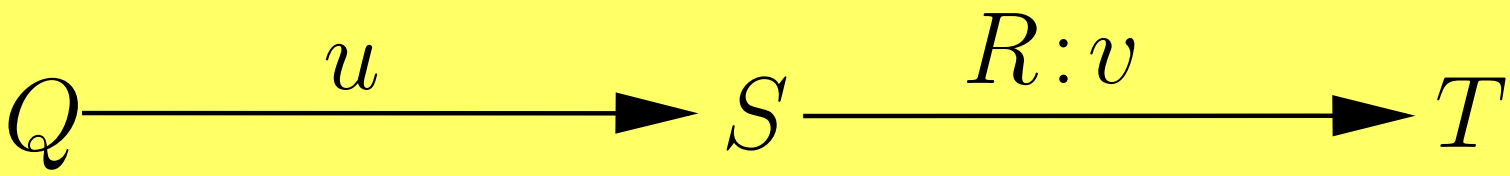
The No-Case Rule



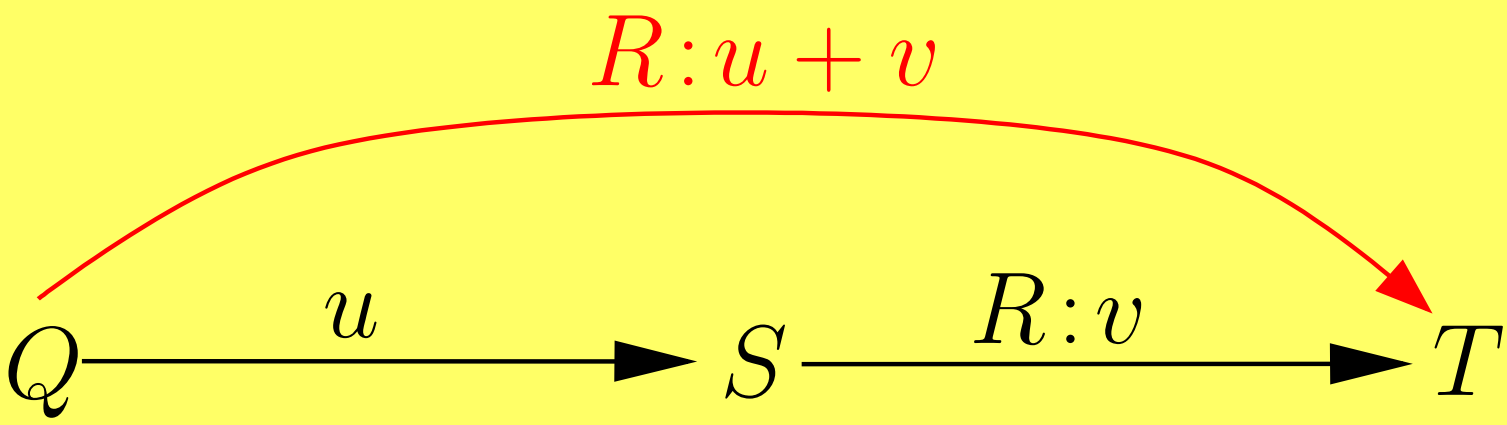
The No-Case Rule



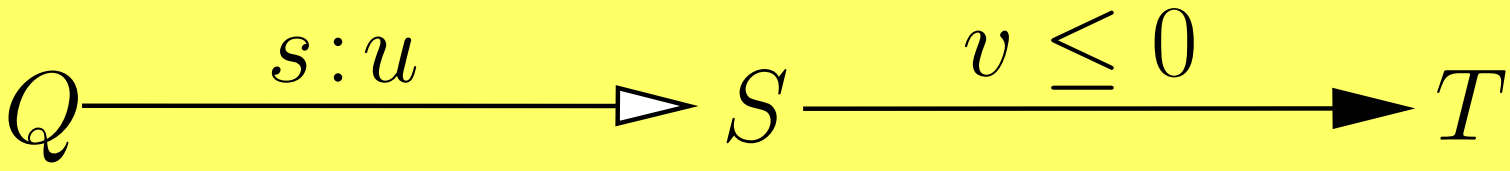
The Upper-Case Rule



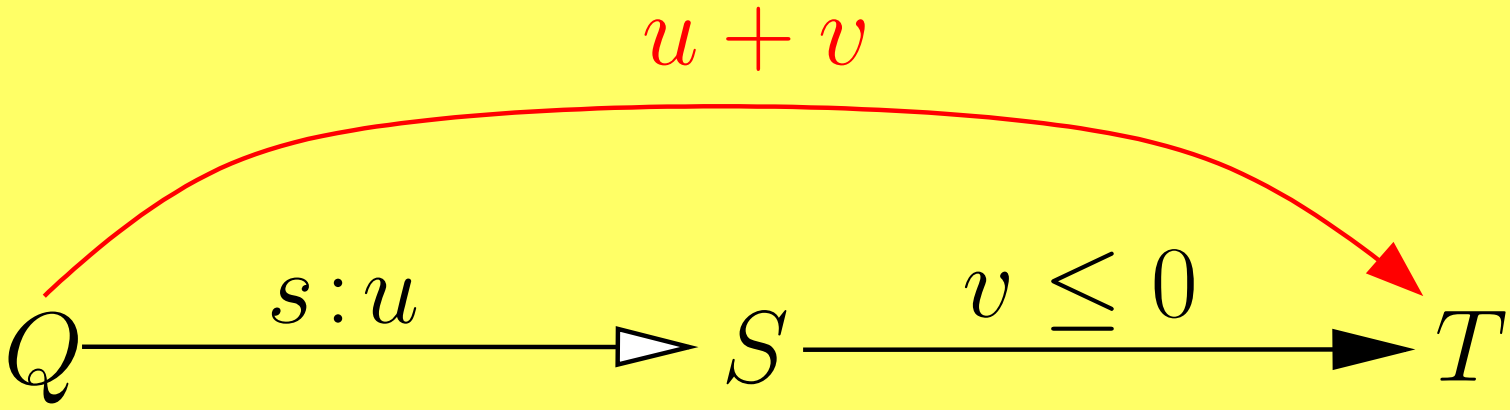
The Upper-Case Rule



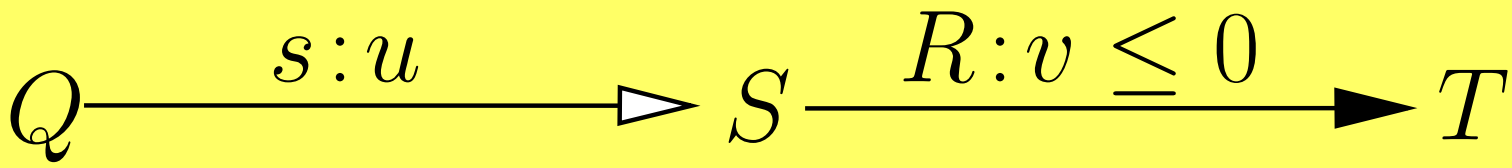
The Lower-Case Rule



The Lower-Case Rule

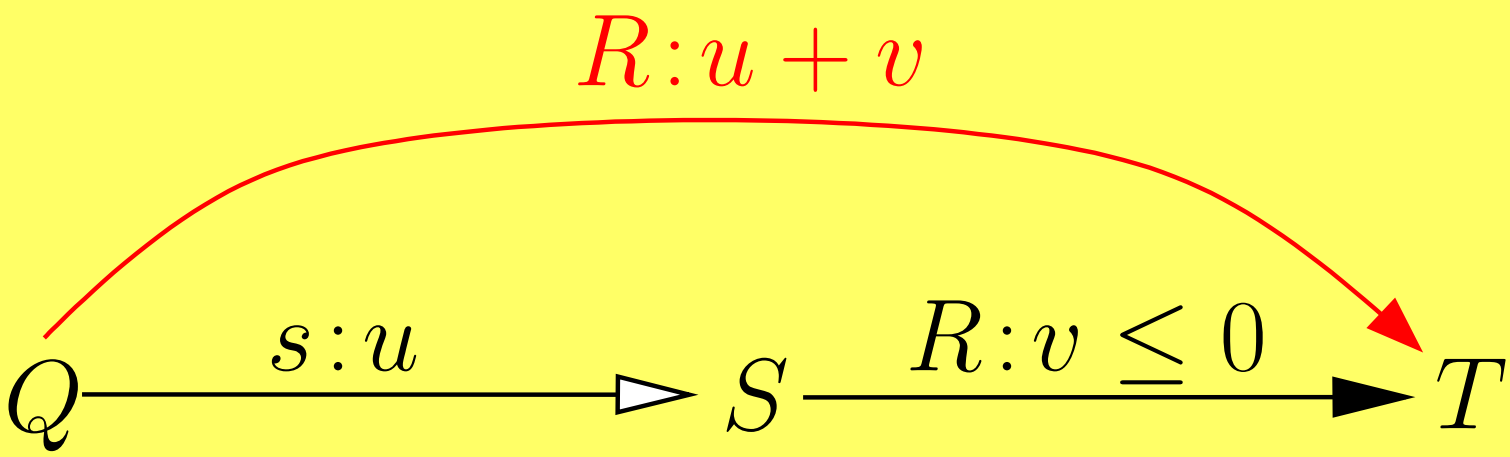


The Cross-Case Rule



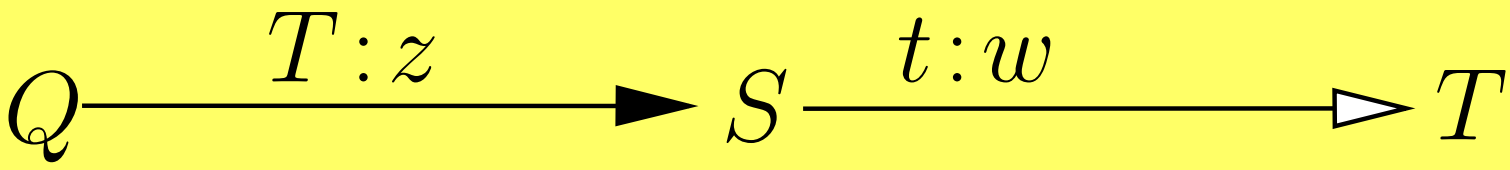
$$R \not\equiv S$$

The Cross-Case Rule



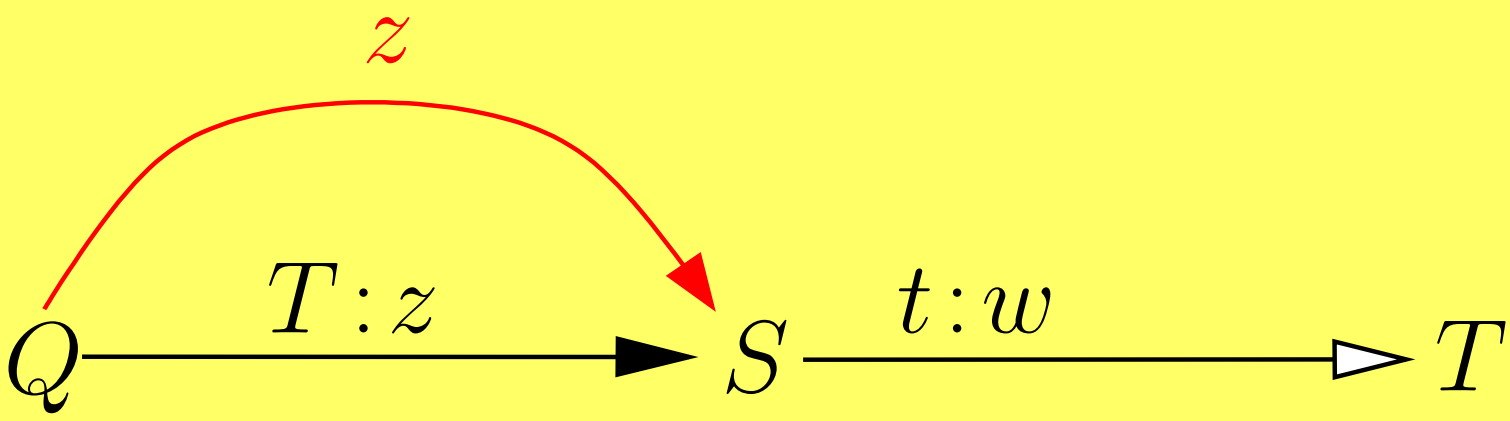
$$R \not\equiv S$$

The Label-Removal Rule



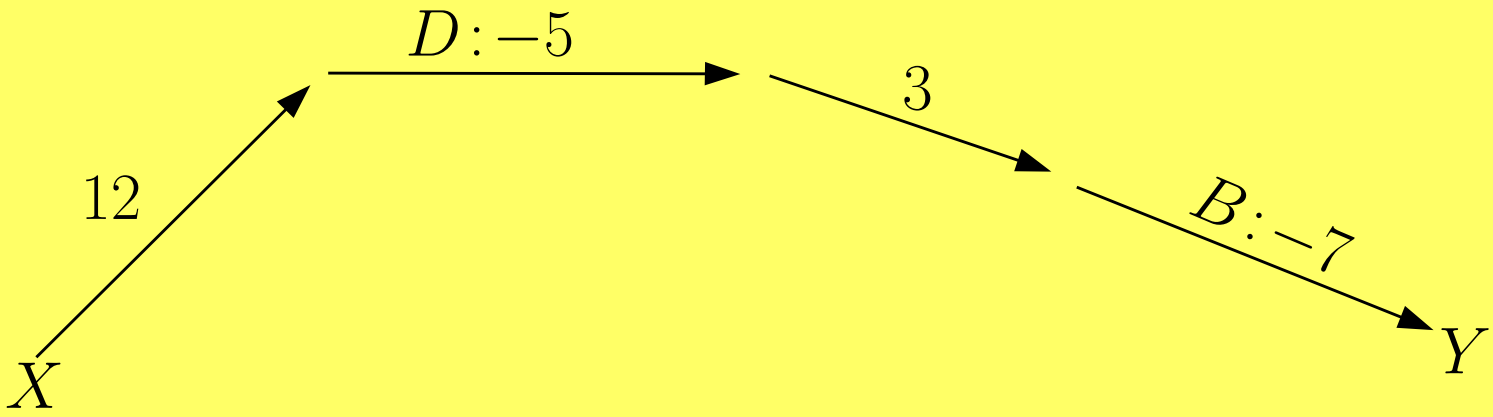
$$z \geq -w$$

The Label-Removal Rule

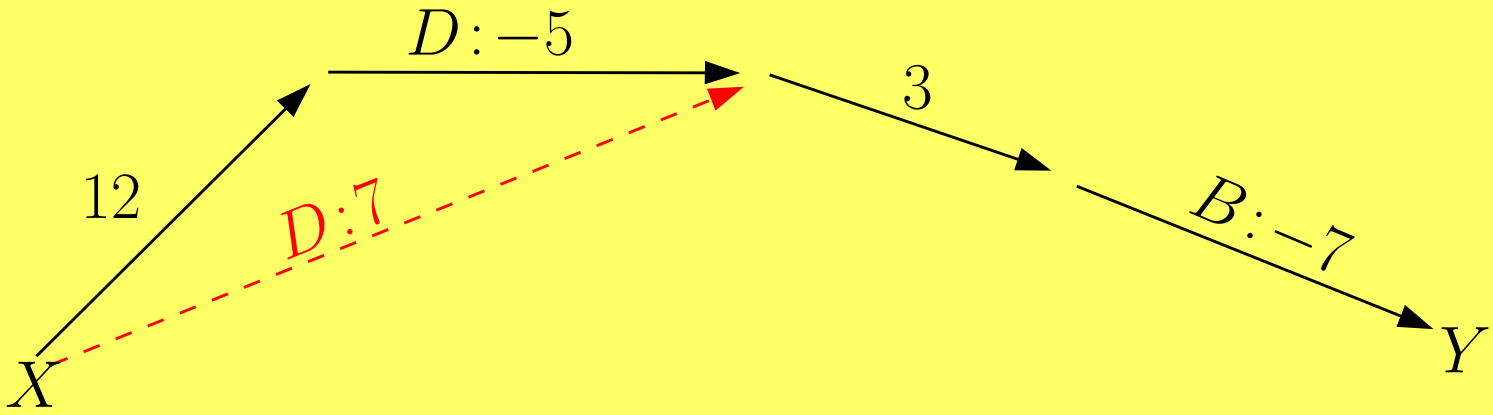


$$z \geq -w$$

Reducing Away Upper-Case Edges

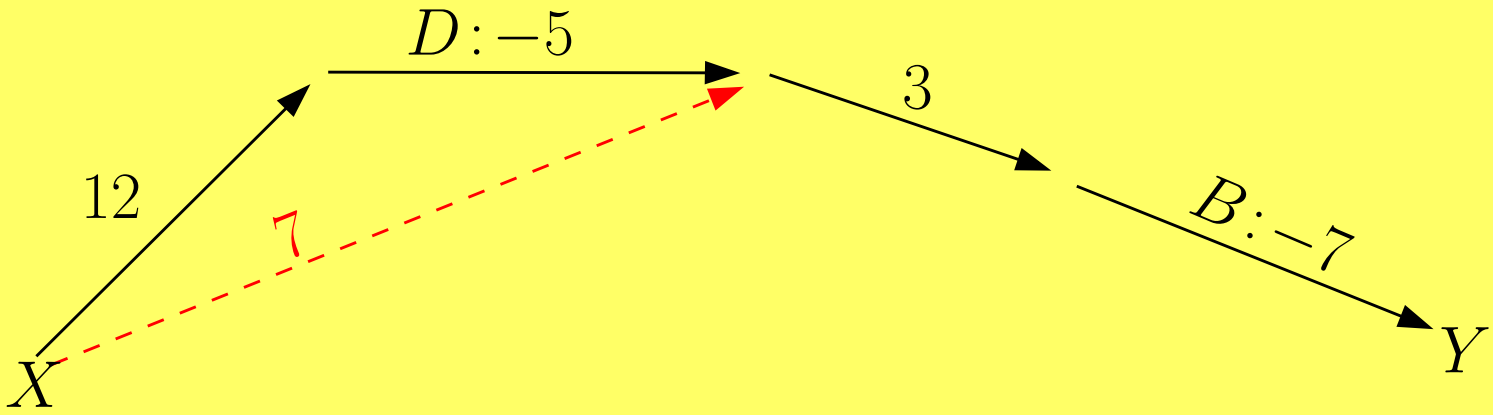


Reducing Away Upper-Case Edges



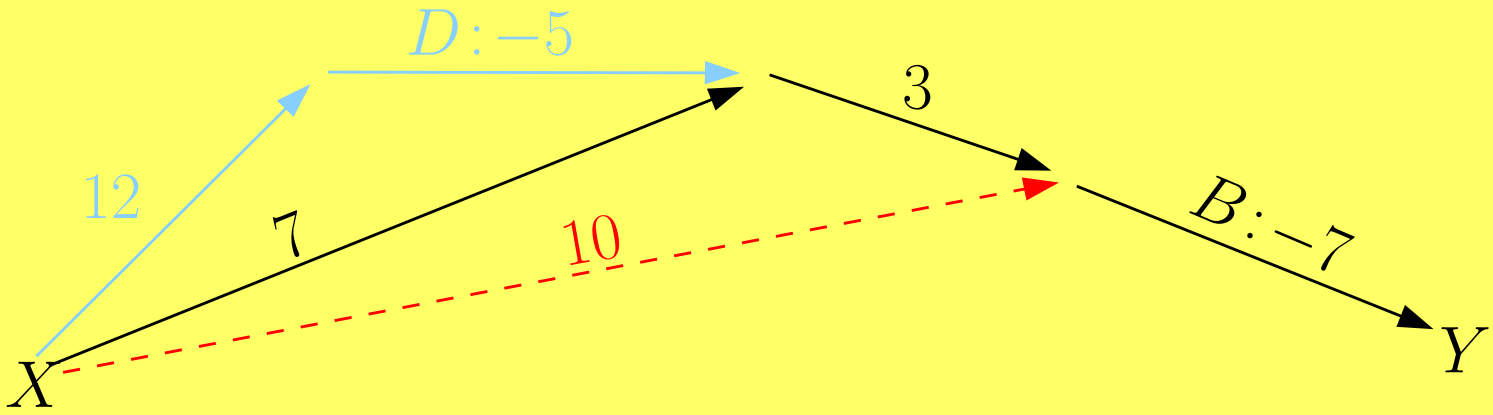
(Upper-Case Rule)

Reducing Away Upper-Case Edges



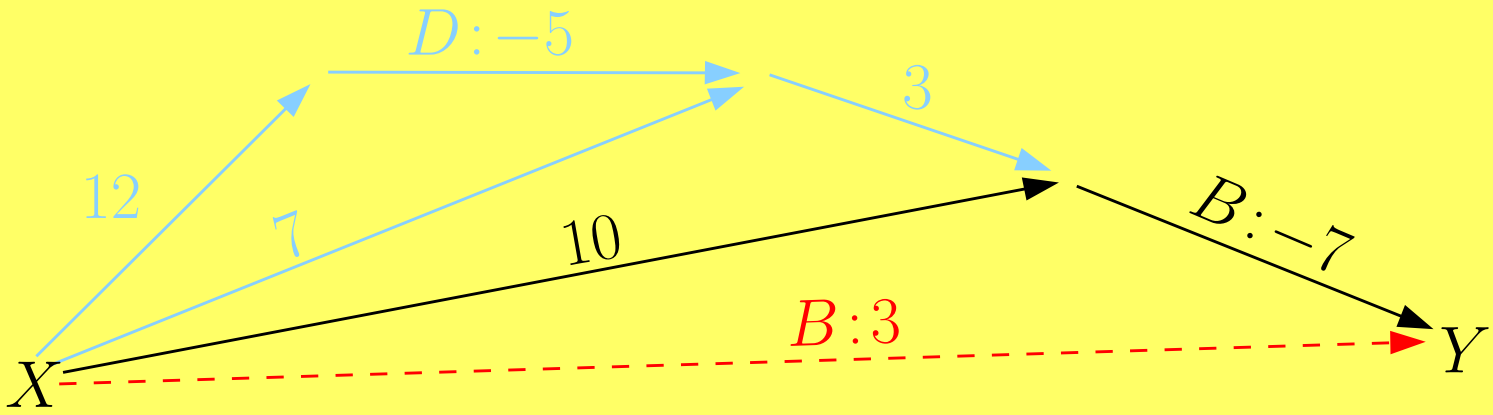
(Label-Removal Rule)

Reducing Away Upper-Case Edges



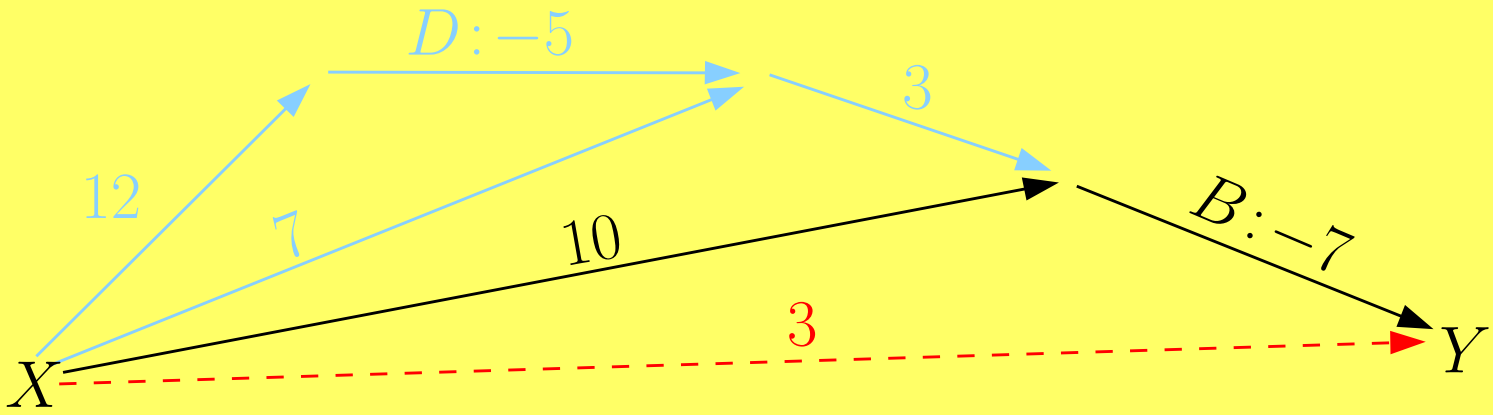
(No-Case Rule)

Reducing Away Upper-Case Edges



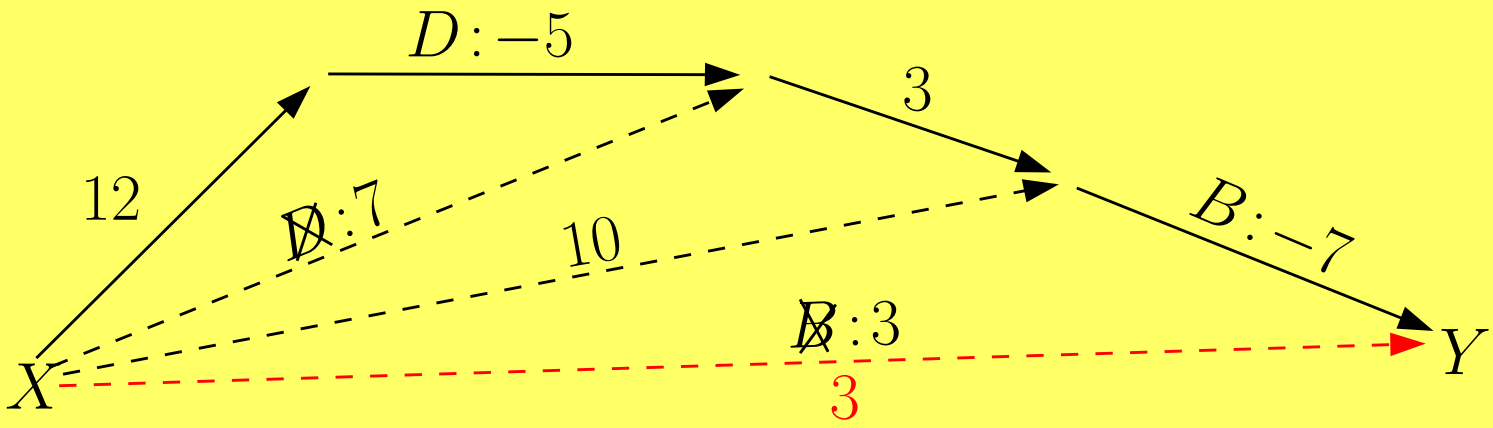
(Upper-Case Rule)

Reducing Away Upper-Case Edges



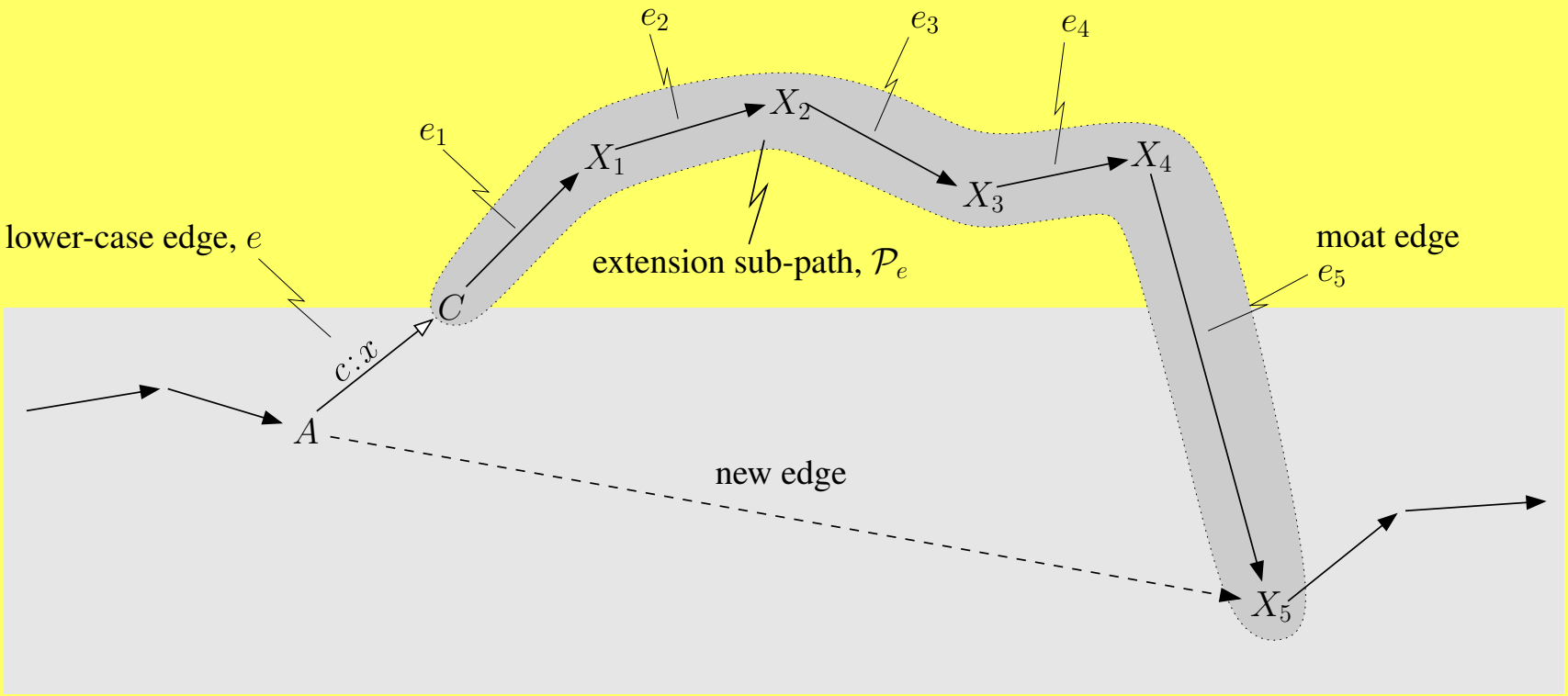
(Label-Removal Rule)

Reducing Away Upper-Case Edges



(Summary)

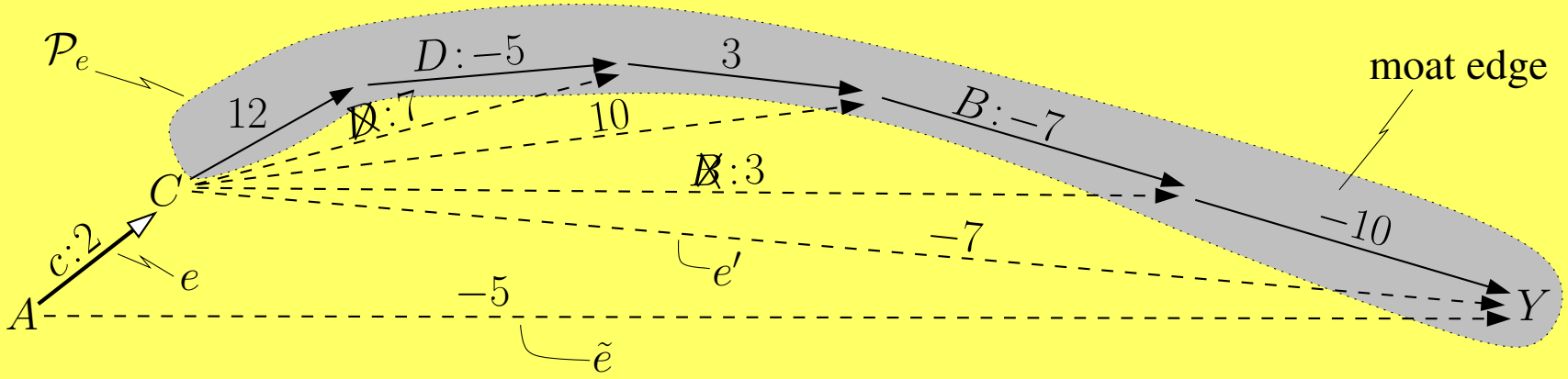
“Reducing Away” a Lower-Case Edge



Note: Moat edge must not have upper-case label C.

Reducing Away a Lower-Case Edge

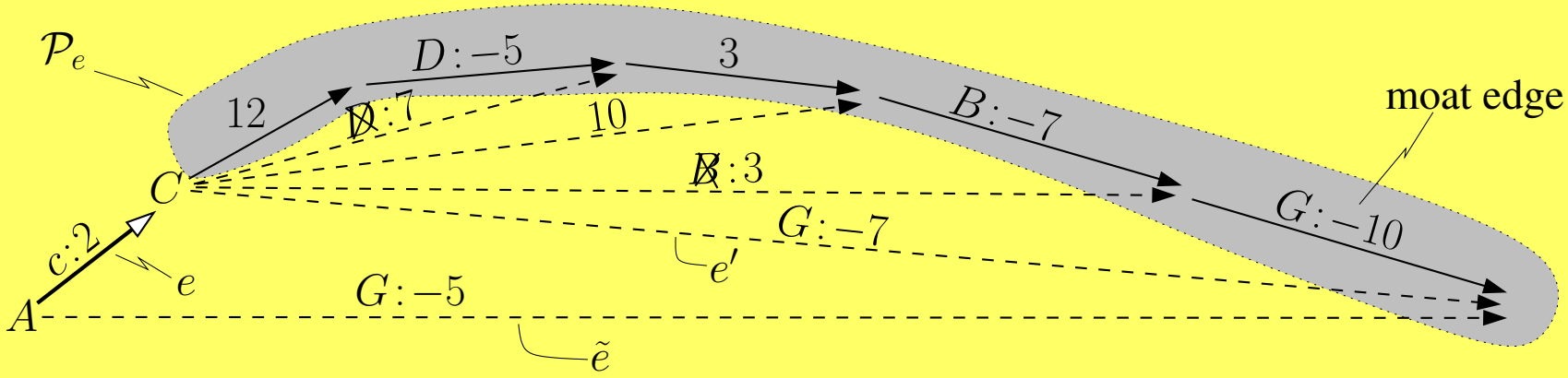
Example 1: “Moat” edge is an ordinary edge:



Generated edge is ordinary, too!

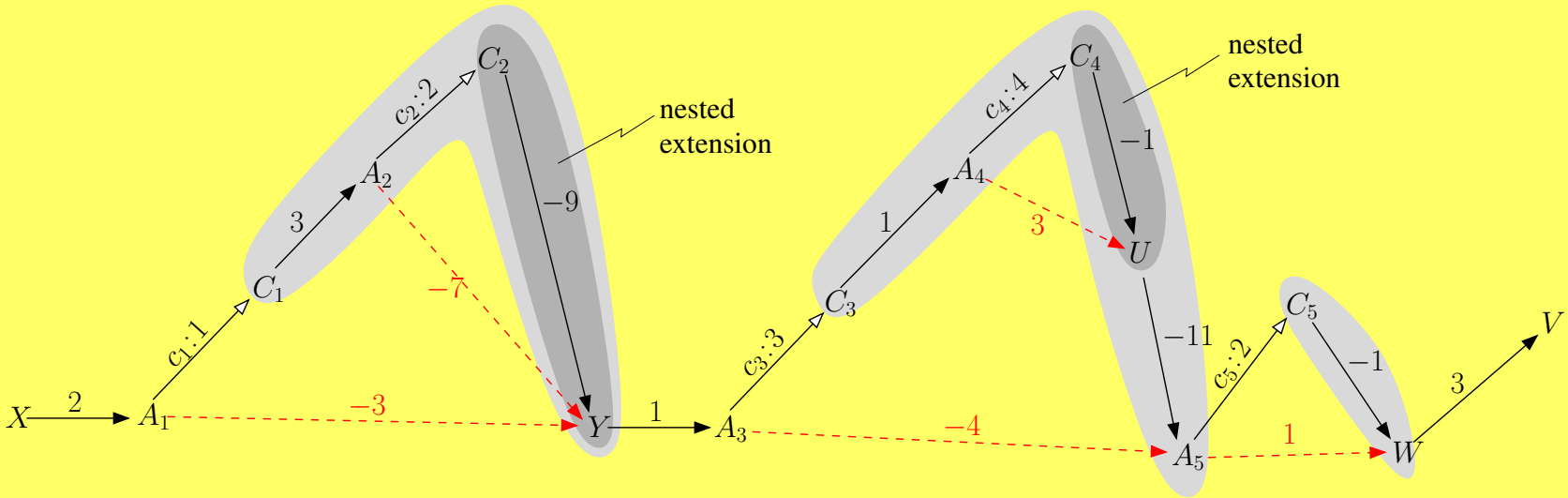
Reducing Away a Lower-Case Edge

Example 2: “Moat” edge is upper-case:



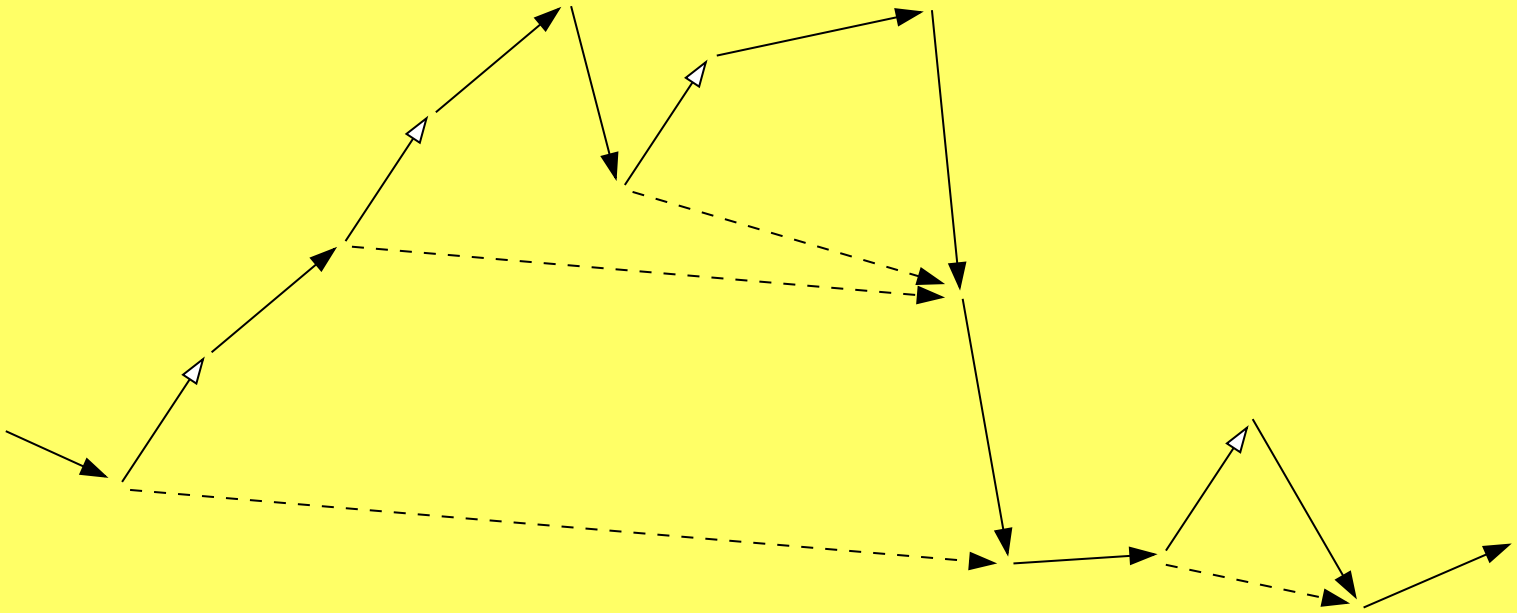
Generated edge is upper-case, too!

Reducing Away LC Edges



Semi-Reducibility

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



Morris' $O(N^4)$-time DC-Checking Alg.

- Searches for paths that can be used to "reduce away" lower-case edges
- Generated edges called *core* edges†
- Original + core edges sufficient to compute $\mathcal{D}^*$ (APSP matrix for semi-reducible paths)†
- STNU DC iff $\mathcal{D}^*$ has non-negative entries along its main diagonal

†(Hunsberger 2010)

Executing STNUs

Executing a DC STNU

- Any DC STNU can be safely executed (by defn: DC $\Rightarrow$ $\exists$ Exec Strategy)
- Time window for (non-contingent) X is:

$$[-\mathcal{D}^*(X, Z),\ \mathcal{D}^*(Z, X)]$$

- Updating $\mathcal{D}^*$ can be done in $O(N^2)$-time per execution event; $O(N^3)$-time overall†

† (Hunsberger 2013a; 2010)

Initial Info for Executing a DC STNU

- $\mathcal{D}^*$ — APSP matrix for semi-reducible paths
- STNU graph — Including all *original* and *core* edges, both *ordinary* and *upper-case*

$\Rightarrow$ Only need to update entries of $\mathcal{D}^*$ that involve Z

Updating $\mathcal{D}^*$ During Execution

- When a non-contingent time-point X is executed, can update $\mathcal{D}^*$ in linear time.
- When a contingent time-point, C, executes, need to *remove* from graph all upper-case edges labeled by C; can update $\mathcal{D}^*$ in $O(N^2)$ time.†
- Total computations: $O(N^3)$ time†
- If updating *all* of $\mathcal{D}^*$, total computations: $O(N^4)$ time.$^\circ$

†(Hunsberger 2010); $^\circ$(Hunsberger 2013a)

Magic Loops in STNUs

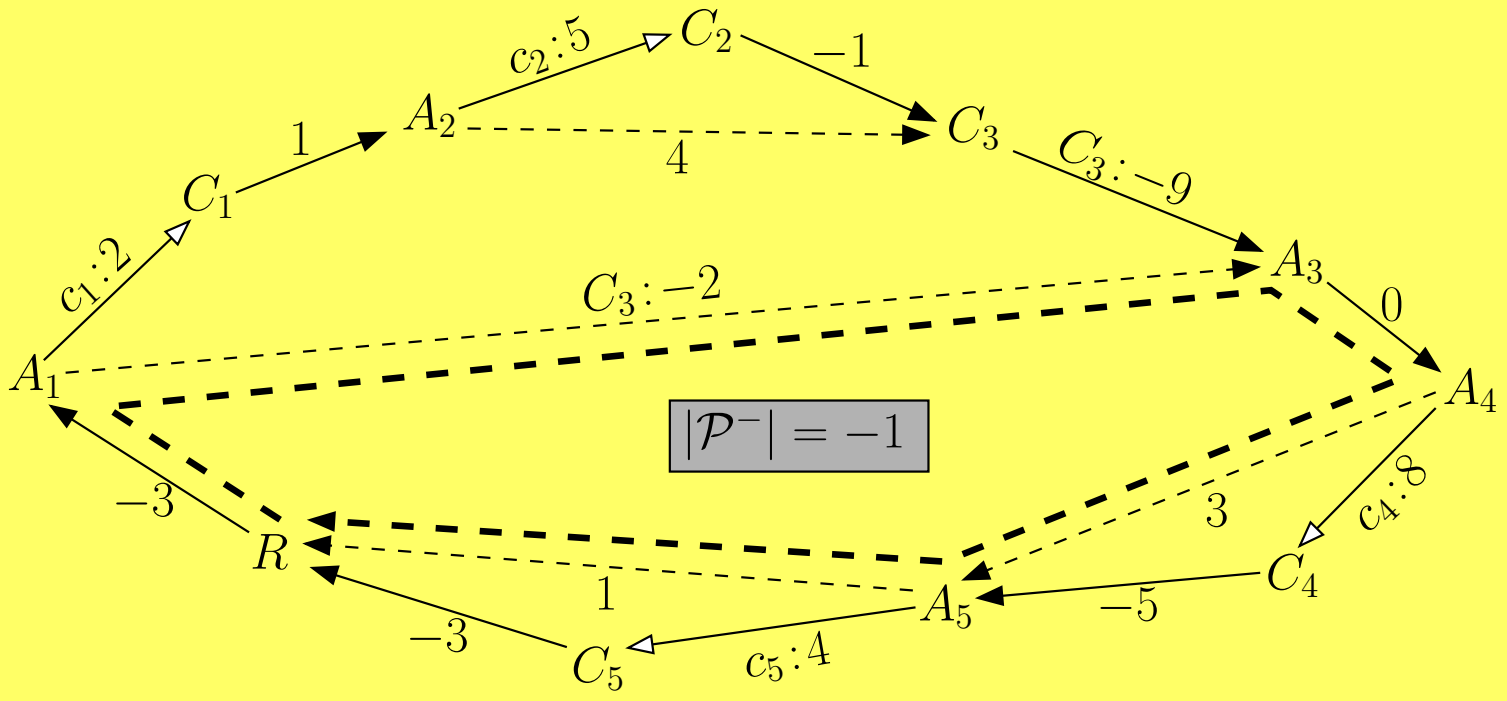
Magic Loops

- Magic loops are complex structures that represent a worst-case scenario for DC-checking algorithms
- Structure of magic loops is bounded
- Exploiting that bound can speed up DC checking for *some* STNUs

(Hunsberger 2013b)

SRN Loop

An SRN loop is a semi-reducible loop having negative unlabeled length.

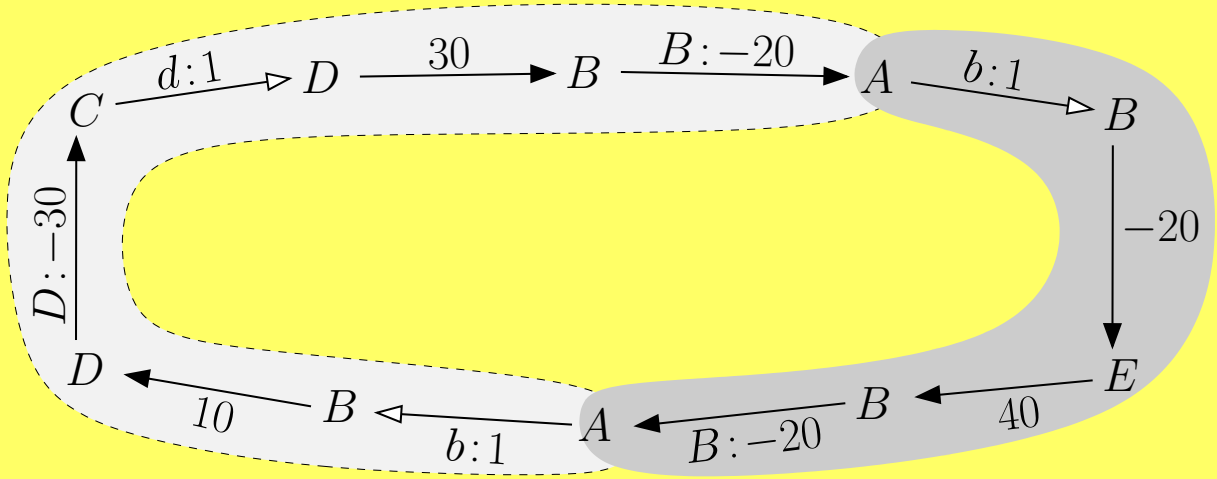


Indivisible SRN Loop (iSRN Loop)

- $\#\mathcal{P} =$ the number of occurrences of lower-case edges in $\mathcal{P}$.
- An SRN loop, $\mathcal{P}$, is indivisible if it has no subsidiary SRN loops, $\mathcal{P}'$, having fewer occurrences of LC edges than $\mathcal{P}$.

Example of an Indivisible SRN Loop

This SRN loop has lots of sub-loops.
But none of the sub-loops are SRN loops.



This sub-loop is not semi-reducible

This sub-loop has non-negative length

New Results

- An iSRN loop can have at most $2^K - 1$ occurrences of lower-case edges.
- A magic loop is an iSRN loop, $\mathcal{P}$, for which $\#\mathcal{P} = 2^K - 1$.
- There are STNUs in which every iSRN loop is a magic loop!

(K is the number of contingent links in the STNU.)

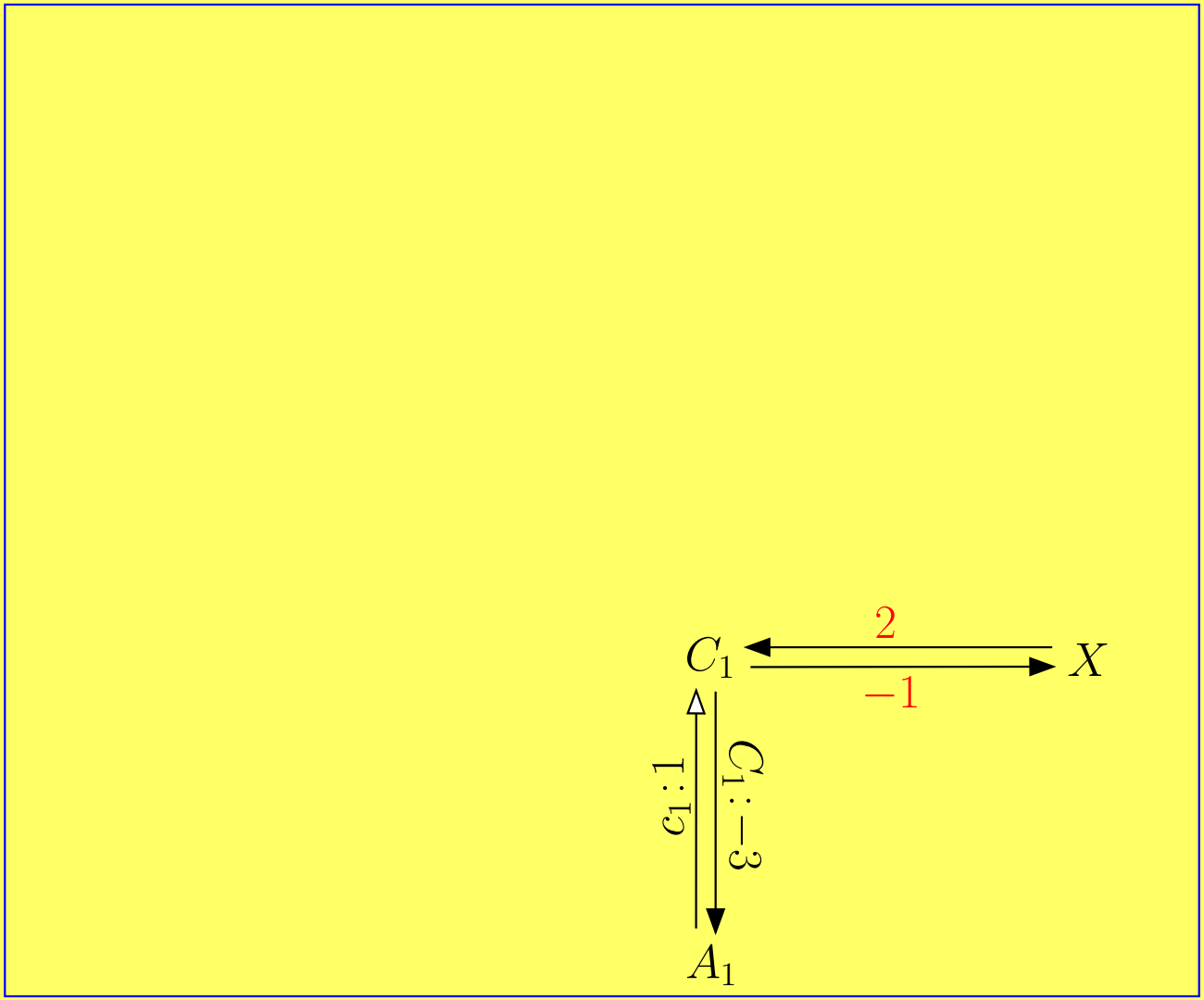
(Hunsberger 2013b)

Pre-Processing Step for DC Checking

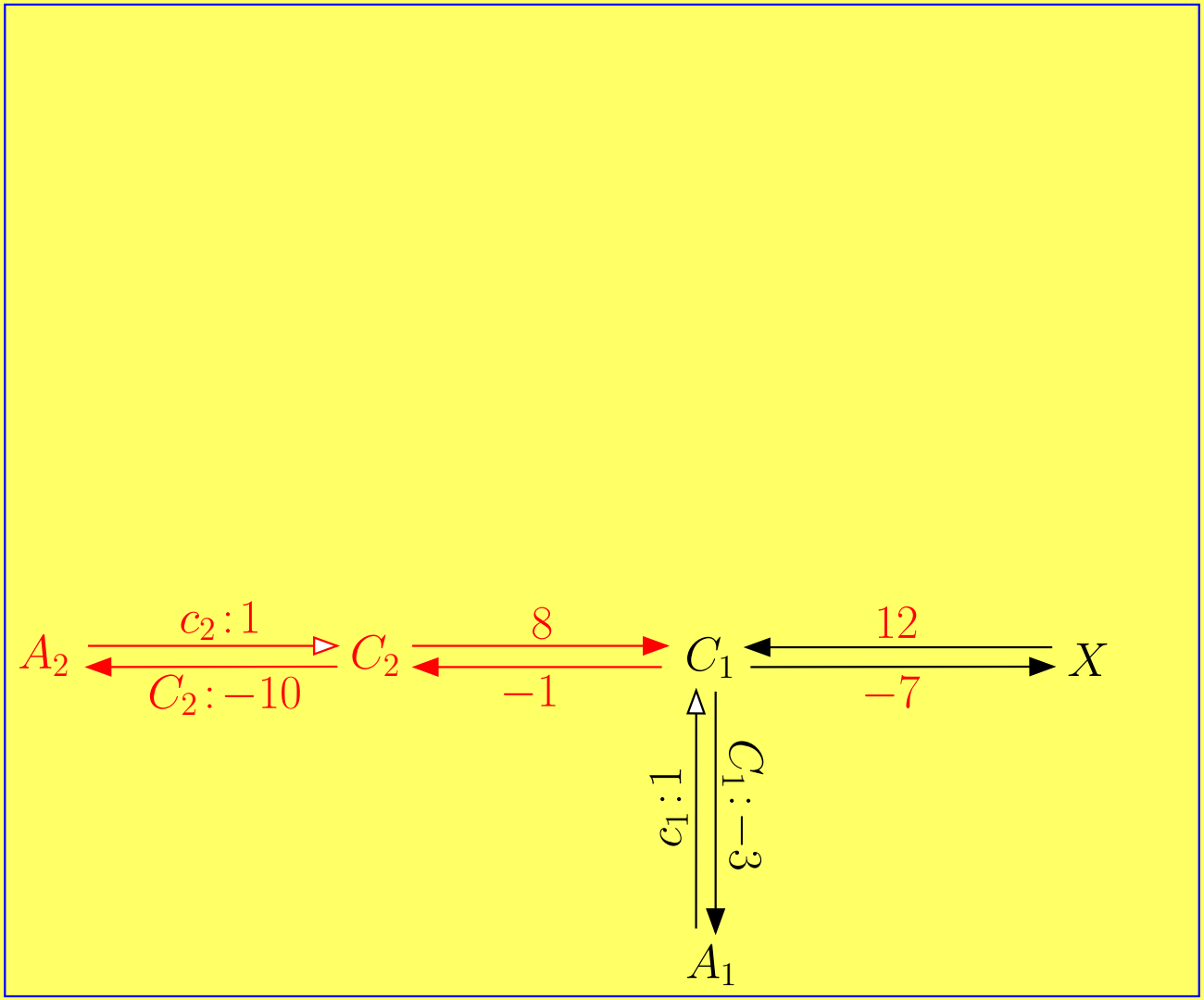
- $O(N^3)$-time pre-processing step
- Exploits bound of $2^K - 1$
- Decreases DC-checking time for some STNUs from $O(N^4)$ to $O(N^3)$
- Does not change worst-case performance

(Hunsberger 2013b)

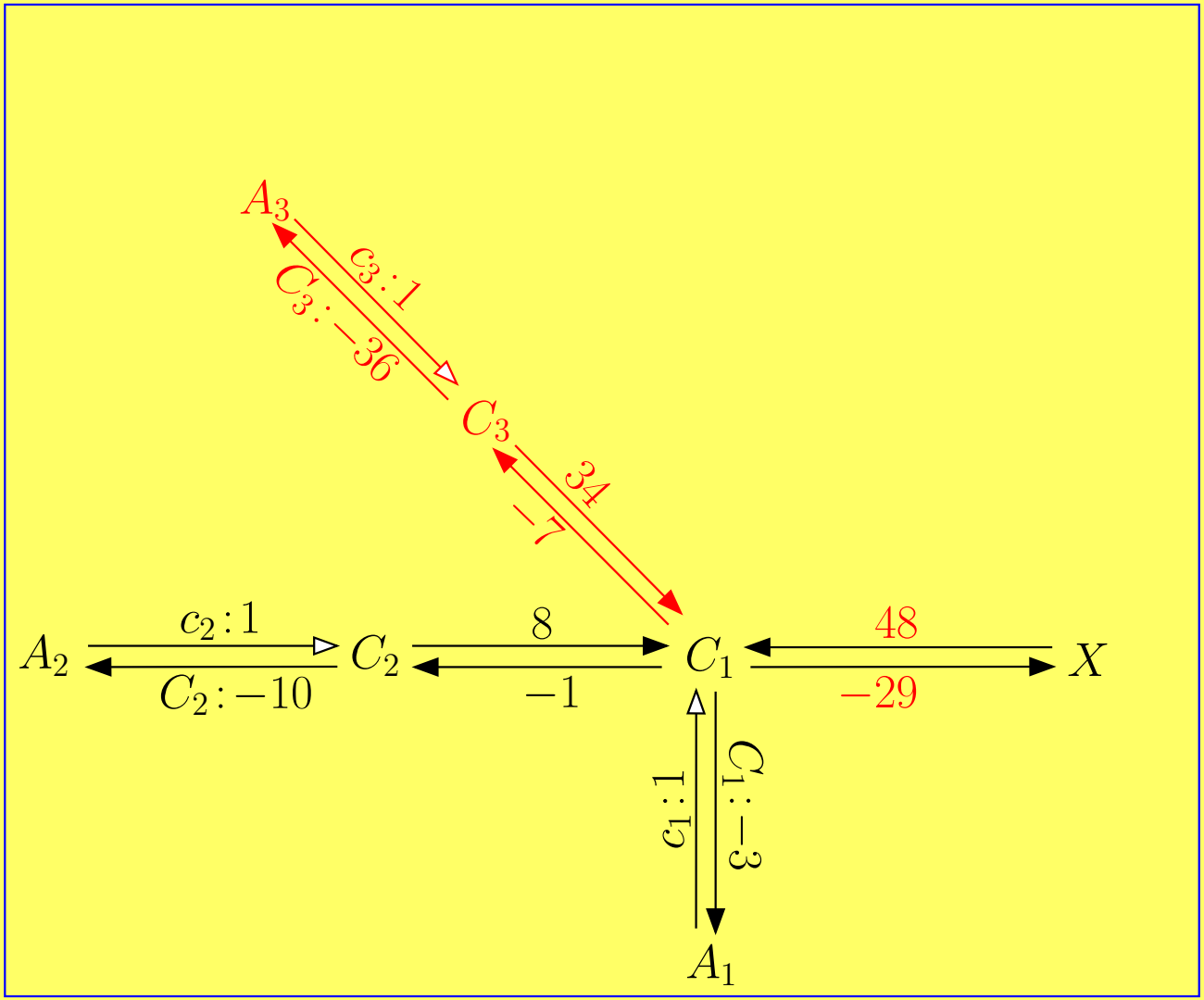
Constructing STNUs with Magic Loops



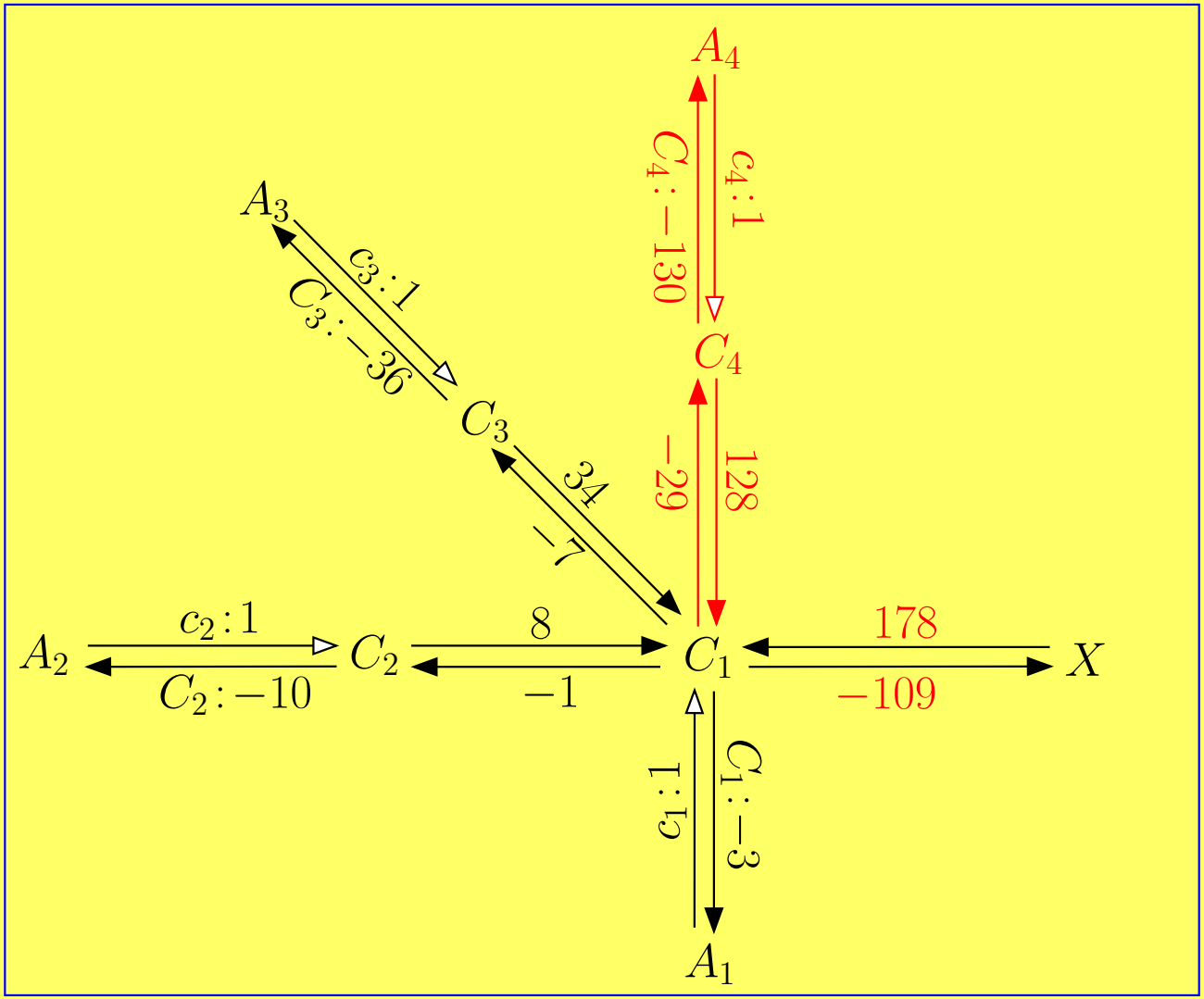
Constructing STNUs with Magic Loops



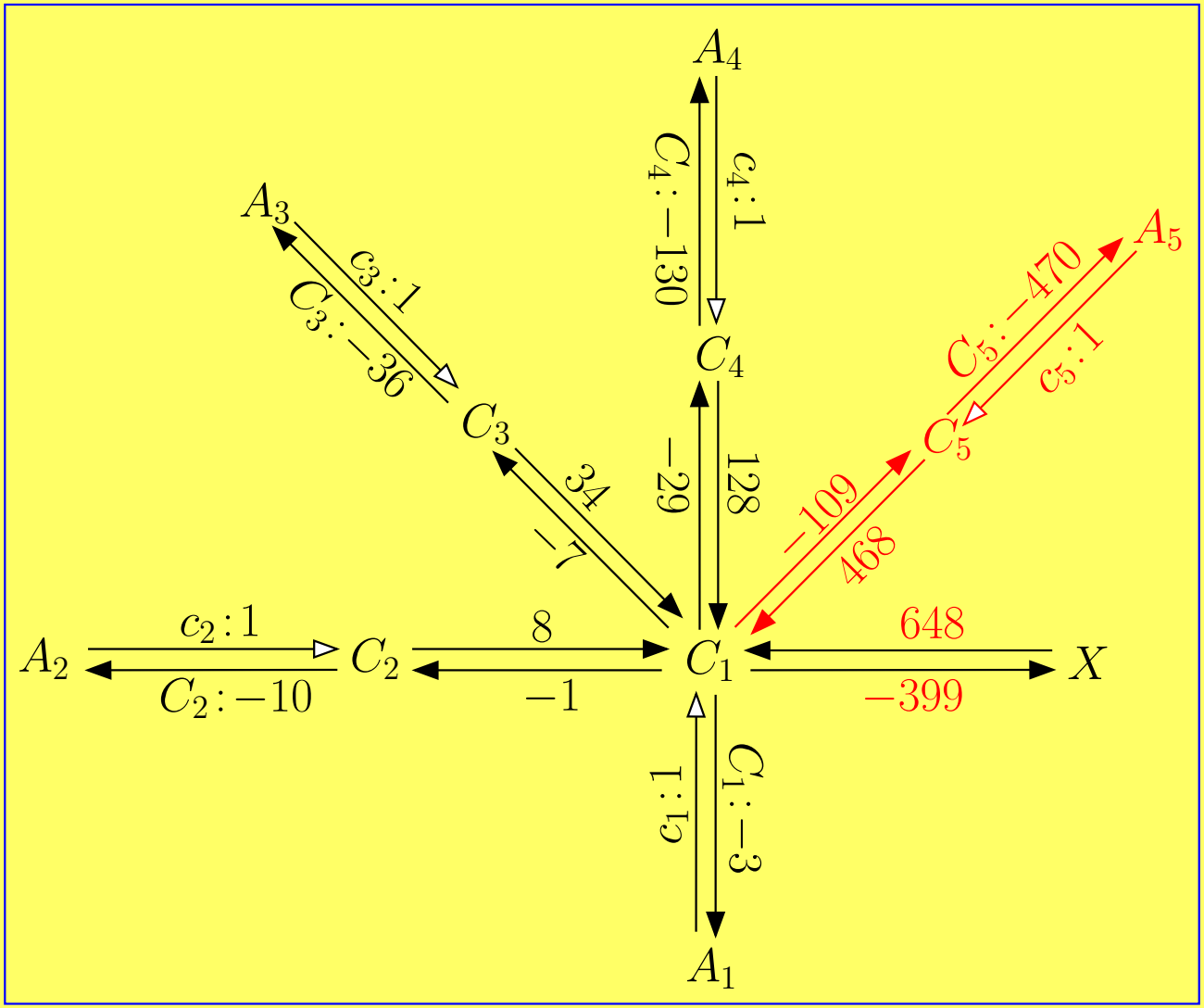
Constructing STNUs with Magic Loops



Constructing STNUs with Magic Loops



Constructing STNUs with Magic Loops



Conclusions

Conclusions

- STNUs can represent temporal constraints among actions with uncertain durations
- $O(N^4)$-time DC-checking algorithm
- $O(N^3)$-time execution algorithm
- Magic loop analysis can speed up some DC checking

Related Work/Future Directions

- Incremental DC-checking algorithms*
- Extensions to accommodate disjunctive choice, uncontrollable branch points, preference, probability†
- Open Question: Can the pool of STNUs for which DC checking can be done in $O(N^3)$ time be expanded?

*(Shah et al. 2007; Stedl and Williams 2005)

†(Effinger et al. 2009; Conrad 2010; Rossi, Venable, and Yorke-Smith 2006; Morris et al. 2005; Venable et al. 2010; Hunsberger, Posenato, and Combi 2012)

STNU-related talk at ICAPS-2013

Incremental Dynamic Controllability Revisited

—Mikael Nilsson, Jonas Kvarnstrm,
and Patrick Doherty

—on Friday!

References

- 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.
- Conrad, P. R. 2010. Flexible execution of plans with choice and uncertainty. Master's thesis, Massachusetts Institute of Technology.
- 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.
- Effinger, R.; Williams, B.; Kelly, G.; and Sheehy, M. 2009. Dynamic controllability of temporally-flexible reactive programs. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 09)*. AAAI Press.
- 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.; Posenato, R.; and Combi, C. 2012. The dynamic controllability of conditional STNs with uncertainty. In *Proceedings of the Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx) Workshop associated with the ICAPS-2012 Conference*, 121–128.
- 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 *Proceedings of the 20th International Symposium on Temporal Representation and Reasoning (TIME-13)*, To Appear.
- 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)*.
- 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, R.; Morris, P.; Khatib, L.; and Yorke-Smith, N. 2005. Temporal constraint reasoning with preferences and probabilities. In Brafman, R., and Junker, U., eds., *Proceedings of the IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, 150–155.

- 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.
- 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.
- 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.
- Rossi, F.; Venable, K. B.; and Yorke-Smith, N. 2006. Uncertainty in soft temporal constraint problems: A general framework and controllability algorithms for the fuzzy case. *Journal of Artificial Intelligence Research* 27:617–674.
- Shah, J.; Stedl, J.; Robertson, P.; and Williams, B. C. 2007. A fast incremental algorithm for maintaining dispatchability of partially controllable plans. In Mark Boddy et al., ed., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*. AAAI Press.
- Stedl, J., and Williams, B. C. 2005. A fast incremental dynamic controllability algorithm. In *Proceedings of the ICAPS Workshop on Plan Execution: A Reality Check*, 69–75.
- 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.
- Venable, K. B.; Volpato, M.; Peintner, B.; and Yorke-Smith, N. 2010. Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In *Proceedings of COPLAS 2010: ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*, 50–59.