Timelines with Temporal Uncertainty

Alessandro Cimatti **Andrea Micheli** Marco Roveri

Embedded Systems Unit
Fondazione Bruno Kessler, Trento, Italy
amicheli@fbk.eu

18th July 2013

AAAI 2013

Outline

Outline

Temporal Planning (With Temporal Uncertainty)

Our setting: Temporal Planning in presence of Temporal Uncertainty, i.e. when some activities cannot be temporally controlled by the plan executor.

Temporal Planning (With Temporal Uncertainty)

Our setting: Temporal Planning in presence of Temporal Uncertainty, i.e. when some activities cannot be temporally controlled by the plan executor.

	Temporal Uncertainty	
	No	Yes
Deciding Activities (Temporal Planning)		
Fixed Activities (Scheduling)		

Temporal Planning (With Temporal Uncertainty)

Our setting: Temporal Planning in presence of Temporal Uncertainty, i.e. when some activities cannot be temporally controlled by the plan executor.

	Temporal Uncertainty	
	No	Yes
Deciding Activities (Temporal Planning)	PDDL 2.1, Timelines	
Fixed Activities (Scheduling)		

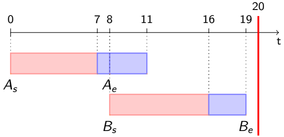
Temporal Planning (With Temporal Uncertainty)

Our setting: Temporal Planning in presence of Temporal Uncertainty, i.e. when some activities cannot be temporally controlled by the plan executor.

	Temporal Uncertainty	
	No	Yes
Deciding Activities (Temporal Planning)	PDDL 2.1, Timelines	Timelines with Temporal Uncertainty
Fixed Activities (Scheduling)		

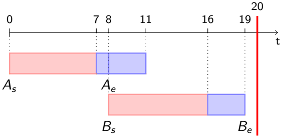
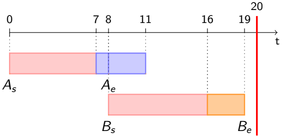
Temporal Planning (With Temporal Uncertainty)

Our setting: Temporal Planning in presence of Temporal Uncertainty, i.e. when some activities cannot be temporally controlled by the plan executor.

	Temporal Uncertainty	
	No	Yes
Deciding Activities (Temporal Planning)	PDDL 2.1, Timelines	Timelines with Temporal Uncertainty
Fixed Activities (Scheduling)		

Temporal Planning (With Temporal Uncertainty)

Our setting: Temporal Planning in presence of Temporal Uncertainty, i.e. when some activities cannot be temporally controlled by the plan executor.

<table>
<tr><td rowspan="2"></td><td colspan="2">Temporal Uncertainty</td></tr>
<tr><td>No</td><td>Yes</td></tr>
<tr><td>Deciding Activities
(Temporal Planning)</td><td>PDDL 2.1, Timelines</td><td>Timelines with
Temporal Uncertainty</td></tr>
<tr><td>Fixed Activities
(Scheduling)</td><td>
</td><td>
</td></tr>
</table>

Timeline Planning

Underlying Idea:
Generate a sequence of **activities** for a set of components according to a *Domain Theory* that fulfill a set of (temporal) constraints.

Timeline Planning

Underlying Idea:
Generate a sequence of **activities** for a set of components according to a *Domain Theory* that fulfill a set of (temporal) constraints.

Planners

- HSTS: Muscettola [1993]
- Europa: Frank and Jónsson [2003]
- APSI: Cesta et al. [2009]
- CNT: Verfaillie et al. [2010]

Timeline Planning

Underlying Idea:
Generate a sequence of **activities** for a set of components according to a *Domain Theory* that fulfill a set of (temporal) constraints.

Planners

- HSTS: Muscettola [1993]
- Europa: Frank and Jónsson [2003]
- APSI: Cesta et al. [2009]
- CNT: Verfaillie et al. [2010]

Applications:
Timeline-based planning is used in many practical applications where temporal constraints are predominant (e.g. Activity Planning & Scheduling for Space Operations).

Contributions

1. Formalization of Timeline Planning with and without Temporal Uncertainty
    - Abstract syntax
    - Problem definition
    - Formal semantics

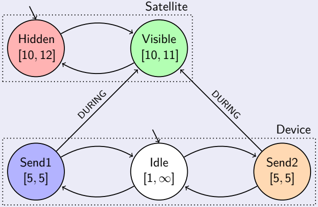
Contributions

1. Formalization of Timeline Planning with and without Temporal Uncertainty
   - Abstract syntax
   - Problem definition
   - Formal semantics
2. Bounded-horizon, strong controllability problem sound and complete encoding in first-order logic.
   - Directly derived from formal semantics
   - APSI-derived concrete syntax
   - Made practical by SMT($\mathcal{LRA}$)

Outline

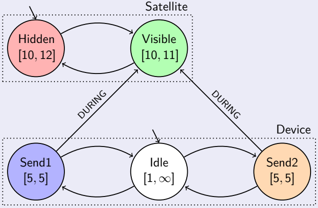
Formalization of Timelines (without Temporal Uncertainty)

Formalization



Formalization of Timelines (without Temporal Uncertainty)

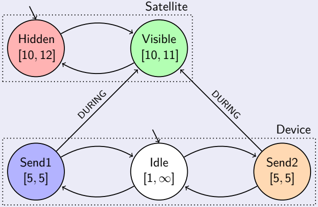
Formalization



- **Generators** describe component behaviors

Formalization of Timelines (without Temporal Uncertainty)

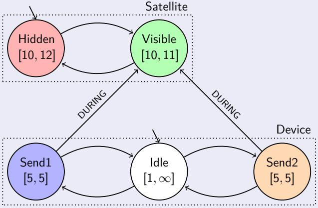
Formalization



- **Generators** describe component behaviors
- **Synchronizations** describe inter-component requirements via *Quantified Allen Relations*

Formalization of Timelines (without Temporal Uncertainty)

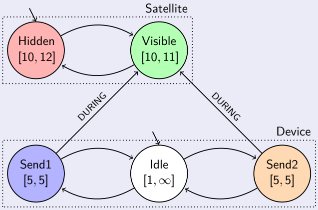
Formalization



- **Generators** describe component behaviors
- **Synchronizations** describe inter-component requirements via *Quantified Allen Relations*
- **Facts** constrain the desired executions (e.g $\textsc{Device.Send2} \in [30, \infty)$)

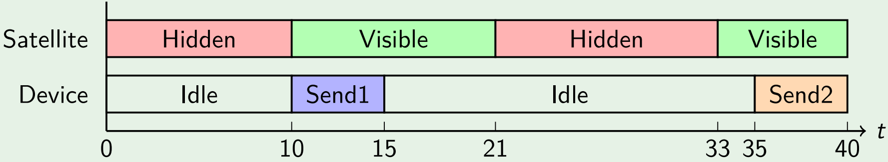
Formalization of Timelines (without Temporal Uncertainty)

Formalization



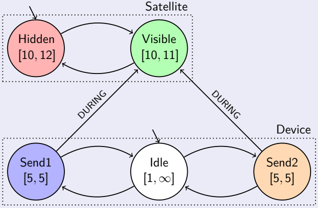
- **Generators** describe component behaviors
- **Synchronizations** describe inter-component requirements via *Quantified Allen Relations*
- **Facts** constrain the desired executions (e.g $\textsc{Device.Send2} \in [30, \infty)$)

Evolution



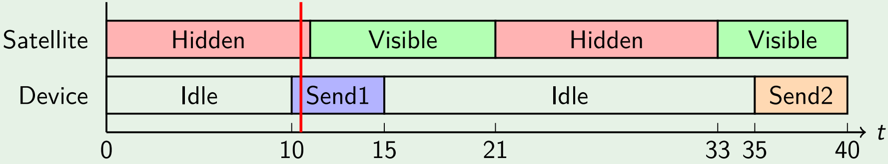
Formalization of Timelines (without Temporal Uncertainty)

Formalization



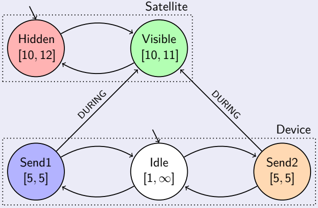
- **Generators** describe component behaviors
- **Synchronizations** describe inter-component requirements via *Quantified Allen Relations*
- **Facts** constrain the desired executions (e.g DEVICE.SEND2 $\in [30, \infty)$)

Evolution



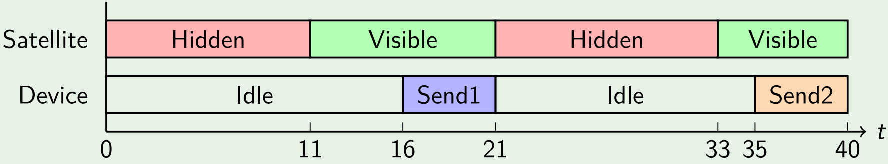
Formalization of Timelines (without Temporal Uncertainty)

Formalization



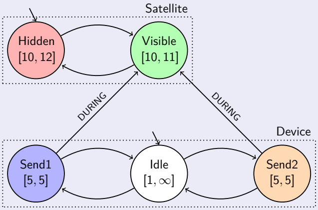
- **Generators** describe component behaviors
- **Synchronizations** describe inter-component requirements via *Quantified Allen Relations*
- **Facts** constrain the desired executions (e.g $\textsc{Device.Send2} \in [30, \infty)$)

Evolution



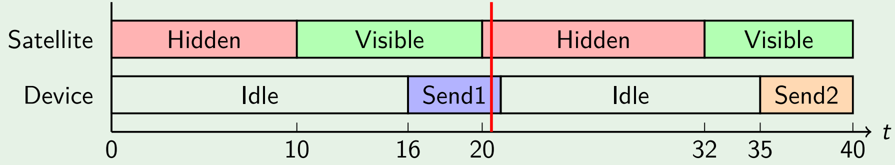
Formalization of Timelines (without Temporal Uncertainty)

Formalization



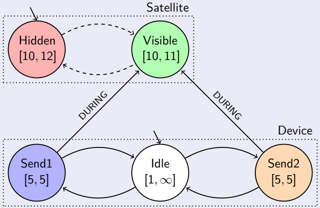
- **Generators** describe component behaviors
- **Synchronizations** describe inter-component requirements via *Quantified Allen Relations*
- **Facts** constrain the desired executions (e.g Device.Send2 $\in [30, \infty)$)

Evolution



Timelines with Temporal Uncertainty

Temporal Uncertainty Annotation

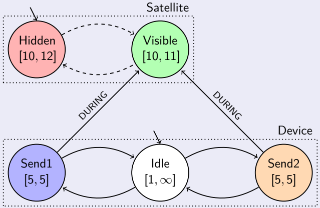


- We *annotate* the domain values with **controllable** or **uncontrollable** flags for both starting and ending time.
- We *annotate* the synchronizations with **contingent** or **free** flag.

Evolution

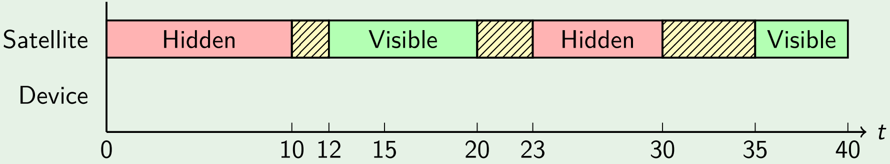
Timelines with Temporal Uncertainty

Temporal Uncertainty Annotation



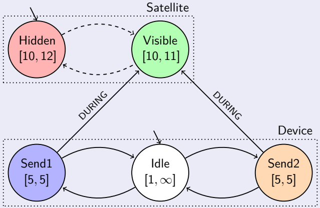
- We *annotate* the domain values with **controllable** or **uncontrollable** flags for both starting and ending time.
- We *annotate* the synchronizations with **contingent** or **free** flag.

Evolution



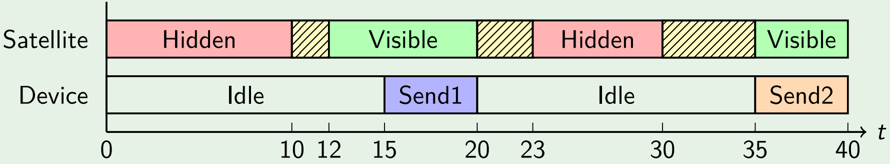
Timelines with Temporal Uncertainty

Temporal Uncertainty Annotation



- We *annotate* the domain values with **controllable** or **uncontrollable** flags for both starting and ending time.
- We *annotate* the synchronizations with **contingent** or **free** flag.

Evolution



Outline

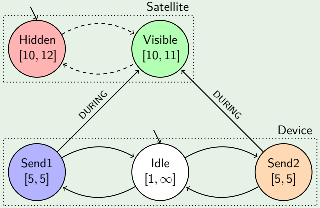
Strong Controllability Bounded-Horizon Encoding

Idea: we assume all durations positive and fix (an upper bound of) the *maximal* number of value changes for each generator withing a given horizon.

Strong Controllability Bounded-Horizon Encoding

Idea: we assume all durations positive and fix (an upper bound of) the *maximal* number of value changes for each generator withing a given horizon.

Example



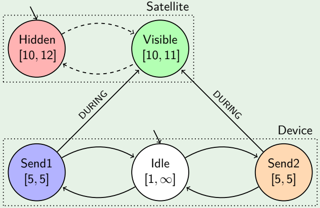
With horizon $H \doteq 240$ we have

- at most 24 values for the Satellite
- at most 80 values for the Device

Strong Controllability Bounded-Horizon Encoding

Idea: we assume all durations positive and fix (an upper bound of) the *maximal* number of value changes for each generator withing a given horizon.

Example



With horizon $H \doteq 240$ we have

- at most 24 values for the Satellite
- at most 80 values for the Device

We can **"unroll"** the problem and we encode it in (**quantified**) First Order Logic modulo the Linear Rational Arithmetic.

Experiments

SMT-Based Implementation

- Implemented on top of the NuSMV model checker
- Fourier-Motzkin Quantifier Elimination to get rid of quantifiers
- MathSAT5 to solve the SMT problems

Experimental Setup

- Three Domains with different problems
- Monolithic vs Incremental implementation
- TO is 1800s, MO is 4Gb

Type	Problem	Monolithic		Incremental	
		Time(s)	Memory(Mb)	Time(s)	Memory(Mb)
Sat	Satellite	6.87	111.5	1.88	31.9
	Machinery1	TO	TO	360.15	611.5
	Meeting	MO	MO	182.52	1897.0
Unsat	Satellite	7.17	126.2	171.25	147.6
	Machinery2	104.86	253.7	113.53	284.4
	Meeting	23.12	630.8	105.17	776.9

Outline

Conclusions

Summary

- Formal description of Timeline Planning with and without Temporal Uncertainty
- Strong Controllability bounded-horizon Planning Problem definition and encoding
- SMT-based prototype of the encoding

Conclusions

Summary

- Formal description of Timeline Planning with and without Temporal Uncertainty
- Strong Controllability bounded-horizon Planning Problem definition and encoding
- SMT-based prototype of the encoding

Future works

- Dynamic and Weak Controllability Planning Problems
- Formalization of resources
- Optimizing Planning: find a solution that minimizes a given cost function
- Competitive implementation

Thanks

Please, come to the poster session for details, explanations and discussion!

Thanks for your attention!

Bibliography

A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and R. Rasconi. The APSI Framework: a Planning and Scheduling Software Development Environment. In *Working Notes of the ICAPS-09 Application Showcase Program*, Thessaloniki, Greece, September 2009.

J. Frank and A.K. Jónsson. Constraint-based Attribute and Interval Planning. *Constraints*, 8(4): 339–364, oct 2003. ISSN 1383-7133 (Print) 1572-9354 (Online).

N. Muscettola. Hsts: Integrating planning and scheduling. Technical report, DTIC Document, 1993.

Gérard Verfaillie, Cédric Pralet, and Michel Lemaître. How to model planning and scheduling problems using constraint networks on timelines. *Knowledge Eng. Review*, 25(3):319–336, 2010.

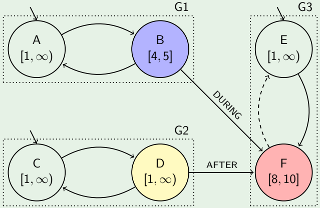
Backup Slides

Backup Slides

Strong Controllability Planning is not Worst Case

One may think that Strong Controllability can be solved by taking the longest or the shortest duration for an activity.

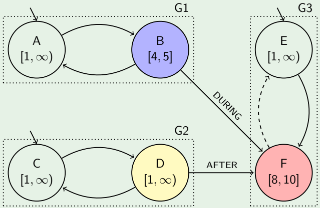
Counterexample

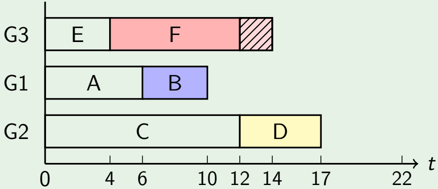


Strong Controllability Planning is not Worst Case

One may think that Strong Controllability can be solved by taking the longest or the shortest duration for an activity.

Counterexample



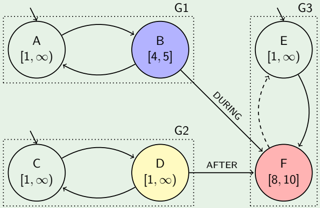


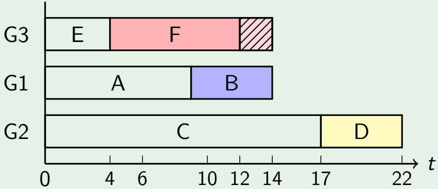
If we take the minimum duration we can violate AFTER constraint

Strong Controllability Planning is not Worst Case

One may think that Strong Controllability can be solved by taking the longest or the shortest duration for an activity.

Counterexample



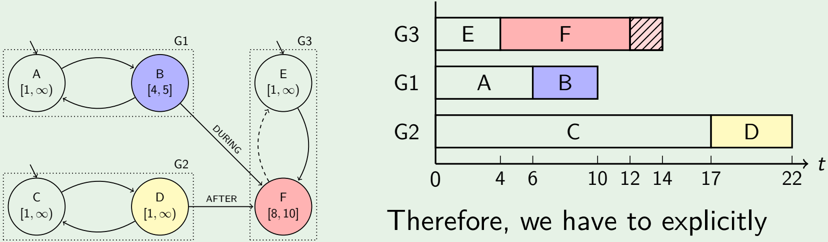


If we take the maximum duration we can violate DURING constraint

Strong Controllability Planning is not Worst Case

One may think that Strong Controllability can be solved by taking the longest or the shortest duration for an activity.

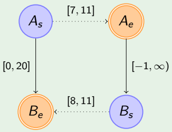
Counterexample



Therefore, we have to explicitly consider temporal uncertainty!

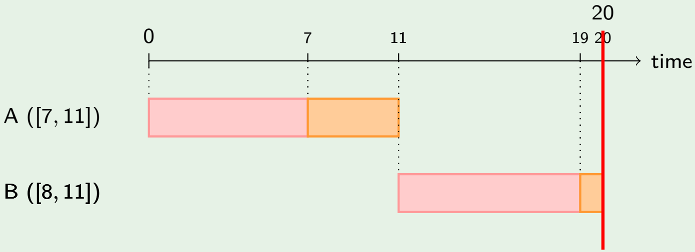
Schedules and Strategies Examples

Example



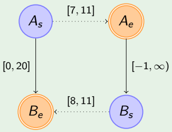
Fixed Schedule
(Strong Controllability)

- $start(A)$ at 0
- $start(B)$ at 11



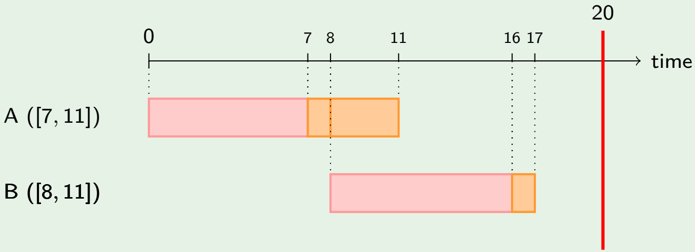
Schedules and Strategies Examples

Example



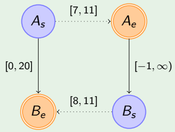
Dynamic Strategy (Dynamic Controllability)

- *start*(A) at 0
- *start*(B) at A_e



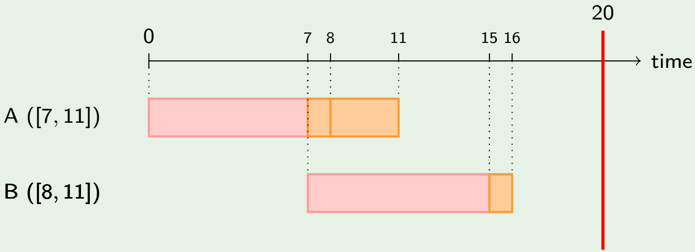
Schedules and Strategies Examples

Example



Clairvoyant Strategy (Weak Controllability)

- *start*(A) at 0
- *start*(B) at $A_e - 1$



Satisfiability Modulo Theory (*SMT*)

SMT is the problem of deciding satisfiability of a first-order Boolean combination of theory atoms in a given theory T.

Given a formula ϕ, ϕ is satisfiable if there exists a model μ such that $\mu \models \phi$.

Satisfiability Modulo Theory (*SMT*)

SMT is the problem of deciding satisfiability of a first-order Boolean combination of theory atoms in a given theory T.

Given a formula ϕ, ϕ is satisfiable if there exists a model μ such that $\mu \models \phi$.

Example

$\phi \doteq (\forall x.(x > 0) \vee (y \geq x)) \wedge (z \geq y)$
is satisfiable in the theory of linear real arithmetic because

$$\mu = \{(y, 6),\ (z, 8)\}$$

is a model that satisfies ϕ.

Satisfiability Modulo Theory (*SMT*)

SMT is the problem of deciding satisfiability of a first-order Boolean combination of theory atoms in a given theory T.

Given a formula ϕ, ϕ is satisfiable if there exists a model μ such that $\mu \models \phi$.

Example

$\phi \doteq (\forall x.(x > 0) \vee (y \geq x)) \wedge (z \geq y)$
is satisfiable in the theory of linear real arithmetic because

$$\mu = \{(y, 6),\ (z, 8)\}$$

is a model that satisfies ϕ.

Theories

Various theories can be used.

In this work:

- $\mathcal{LRA}$ (*Linear Real Arithmetic*)
- $\mathcal{QF\_LRA}$ (*Quantifier-Free Linear Real Arithmetic*)

Quantifier Elimination

Quantifier Elimination Definition

A theory T has quantifier elimination if for every formula Φ, there exists another formula Φ_{QF} without quantifiers which is *equivalent* to it (modulo the theory T)

Quantifier Elimination

Quantifier Elimination Definition

A theory T has quantifier elimination if for every formula Φ, there exists another formula Φ_{QF} without quantifiers which is *equivalent* to it (modulo the theory T)

Quantifier Elimination for $\mathcal{LRA}$

$\mathcal{LRA}$ theory admits quantifier elimination, but elimination algorithms are very costly (doubly exponential in the size of the original formula).

$(\exists x.(x \geq 2y + z) \wedge (x \leq 3z + 5)) \leftrightarrow (2y - 2z - 5 \leq 0)$

Different techniques exists:

- Fourier-Motzkin
- Loos-Weisspfenning
- ...

Quantifier Elimination for $\mathcal{LRA}$

Various techniques

- Fourier-Motzkin
- Loos-Weisspfenning
- ...

Fourier-Motzkin Elimination

- Procedure that eliminates a variable from a **conjunction** of linear inequalities.
- It can be applied to a general $\mathcal{LRA}$ formula by computing the DNF and applying the technique to each disjunct.
- The complexity is doubly exponential: in the number of variable to quantify and in the size of the DNF formula.

Fourier-Motzkin Elimination

Let $\psi \doteq \exists x_r . \bigwedge_{i=0}^{N} \sum_{k=1}^{M} a_{ik} x_k \leq b_i$ be the problem we want to solve, where x_r is the variable to eliminate.

We have three kinds of inequalities in a system of linear inequalities:

- $x_r \geq A_h$, where $A_h \doteq b_i - \sum_{k=1}^{r_i-1} a_{ik} x_k$, for $h \in [1, H_A]$
- $x_r \leq B_h$, where $B_h \doteq b_i - \sum_{k=1}^{r_i-1} a_{ik} x_k$, for $h \in [1, H_B]$
- Inequalities in which x_r has no role. Let ϕ be the conjunction of those inequalities.

The system is **equivalent** to $(max_{h=1}^{H_A}(A_h) \leq x_r \leq min_{h=1}^{H_b}(B_h)) \wedge \phi$ and to $(max_{h=1}^{H_A}(A_h) \leq min_{h=1}^{H_b}(B_h)) \wedge \phi$

max and *min* are not linear functions, but we can mimic the formula by using a quadratic number of linear inequalities:

$$\psi \Leftrightarrow (\bigwedge_{i=0}^{H_A} \bigwedge_{j=0}^{H_B} A_i \leq B_j) \wedge \phi$$

Fourier-Motzkin Example

Fourier Motzkin Example: Step 1

Let $\psi \doteq \forall z.((z \geq 4) \rightarrow ((x < z) \wedge (y < z)))$.

We convert all the quantifiers in existentials and we compute the DNF of the quantified part of the formula.

$\psi \Leftrightarrow \neg \exists z.((z \geq 4) \wedge \neg((x < z) \wedge (y < z)))$

$\psi \Leftrightarrow \neg \exists z.((z \geq 4) \wedge (\neg(x < z) \vee \neg(y < z)))$

$\psi \Leftrightarrow \neg \exists z.(((z \geq 4) \wedge \neg(x < z)) \vee ((z \geq 4) \wedge \neg(y < z)))$

Fourier Motzkin Example: Step 2

For every disjunct, we apply the Fourier-Motzkin Elimination:

$((z \geq 4) \wedge (z \leq x)) \Leftrightarrow (4 \leq x)$

$((z \geq 4) \wedge (z \leq y)) \Leftrightarrow (4 \leq y)$

Then, we rebuild the formula:

$\psi \Leftrightarrow \neg((4 \leq x) \vee (4 \leq y))$

$\psi \Leftrightarrow ((x < 4) \wedge (y < 4))$

Temporal Uncertainty Characterization

Temporal Uncertainty can be seen as a **game** between an *Executor* and the adversarial *Nature*.

Temporal Uncertainty Characterization

Temporal Uncertainty can be seen as a **game** between an *Executor* and the adversarial *Nature*.

Rules

- The *Executor* schedules a set of **Controllable Time Points (X_c)**

Temporal Uncertainty Characterization

Temporal Uncertainty can be seen as a **game** between an *Executor* and the adversarial *Nature*.

Rules

- The *Executor* schedules a set of **Controllable Time Points** (X_c)
- The *Executor* must fulfill a set of temporal constraints called **Free Constraints** (C_f)

Temporal Uncertainty Characterization

Temporal Uncertainty can be seen as a **game** between an *Executor* and the adversarial *Nature*.

Rules

- The *Executor* schedules a set of **Controllable Time Points (X_c)**
- The *Executor* must fulfill a set of temporal constraints called **Free Constraints (C_f)**
- The *Nature* tries to prevent the success of the executor scheduling a set of **Uncontrollable Time Points (X_u)**

Temporal Uncertainty Characterization

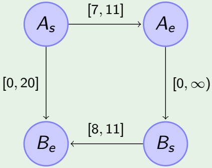
Temporal Uncertainty can be seen as a **game** between an *Executor* and the adversarial *Nature*.

Rules

- The *Executor* schedules a set of **Controllable Time Points (X_c)**
- The *Executor* must fulfill a set of temporal constraints called **Free Constraints (C_f)**
- The *Nature* tries to prevent the success of the executor scheduling a set of **Uncontrollable Time Points (X_u)**
- The *Nature* must fulfill a set of temporal constraints called **Contingent Constraints (C_c)**

Temporal Problems (with Temporal Uncertainty)

Temporal Problems

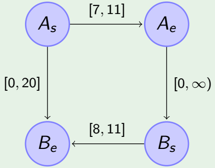


A_s, A_e, B_s, B_e are **Time Points (X_c)**

$\longrightarrow$ represents **Free Constraints (C_f)**

Temporal Problems (with Temporal Uncertainty)

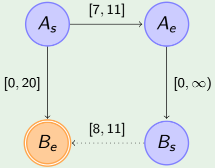
Temporal Problems



A_s, A_e, B_s, B_e are **Time Points (X_c)**

$\longrightarrow$ represents **Free Constraints (C_f)**

Temporal Problems with Uncertainty



A_s, A_e, B_s are **Controllable Time Points (X_c)**
B_e is an **Uncontrollable Time Point (X_u)**

$\longrightarrow$ represents **Free Constraints (C_f)**
$\cdots\rightarrow$ represents **Contingent Constraints (C_c)**

Controllability Levels

Controllability Levels

- **Strong Controllability (No observation)**
  Find a **fixed schedule** for controllable time points

Fixed Schedule

- $start(A)$ at 0
- $start(B)$ at 11

Controllability Levels

- **Strong Controllability (No observation)**
  Find a **fixed schedule** for controllable time points

- **Dynamic Controllability (Past observation)**
  Find a **strategy that depends on past observations only**, for scheduling controllable time points

Fixed Schedule

- $start(A)$ at 0
- $start(B)$ at 11

Dynamic Strategy

- $start(A)$ at 0
- $start(B)$ at C

Controllability Levels

- **Strong Controllability (No observation)**
  Find a **fixed schedule** for controllable time points

- **Dynamic Controllability (Past observation)**
  Find a **strategy that depends on past observations only**, for scheduling controllable time points

- **Weak Controllability (Full observation)**
  Find a **"clairvoyant" strategy** for scheduling controllable time points

Fixed Schedule

- $start(A)$ at 0
- $start(B)$ at 11

Dynamic Strategy

- $start(A)$ at 0
- $start(B)$ at C

Clairvoyant Strategy

- $start(A)$ at 0
- $start(B)$ at $C - 1$