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

1 Introduction

2 Timelines with Temporal Uncertainty

3 Strong Controllability Bounded-Horizon Encoding

4 Conclusion
Outline

1 Introduction

2 Timelines with Temporal Uncertainty

3 Strong Controllability Bounded-Horizon Encoding

4 Conclusion
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.

<table>
<thead>
<tr>
<th>Temporal Uncertainty</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deciding Activities (Temporal Planning)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Activities (Scheduling)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Temporal Uncertainty</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deciding Activities</td>
<td>PDDL 2.1, Timelines</td>
<td></td>
</tr>
<tr>
<td>(Temporal Planning)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed Activities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Scheduling)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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>
<thead>
<tr>
<th>Temporal Uncertainty</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>
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>
<thead>
<tr>
<th>Temporal Uncertainty</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

\[
\begin{array}{cccc}
0 & 7 & 8 & 11 & 16 & 19 & 20 \\
A_s & A_e & B_s & B_e & t & & \\
\end{array}
\]
**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>
<thead>
<tr>
<th>Temporal Uncertainty</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deciding Activities</strong></td>
<td>PDDL 2.1, Timelines</td>
<td><strong>Timelines with Temporal Uncertainty</strong></td>
</tr>
<tr>
<td><strong>Fixed Activities</strong></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>(Scheduling)</td>
<td></td>
<td></td>
</tr>
</tbody>
</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

1. Introduction

2. Timelines with Temporal Uncertainty

3. Strong Controllability Bounded-Horizon Encoding

4. Conclusion
Formalization of Timelines (without Temporal Uncertainty)

**Formalization**

Visible \([10, 11]\)  
Hidden \([10, 12]\)

Send1 \([5, 5]\)  
DURING  
Idle \([1, \infty]\)

Satellite

DURING

Send2 \([5, 5]\)  
Device

Evolution

Hidden  
Visible  
Hidden  
Visible  
Hidden  
Visible  
Hidden  
Visible  
Hidden  
Visible  
Hidden  
Visible  
Hidden  
Visible
Formalization of Timelines (without Temporal Uncertainty)

Formalization

- **Generators** describe component behaviors
Formalization of Timelines (without Temporal Uncertainty)

Visible $[10, 11]$

Hidden $[10, 12]$

Send1 $[5, 5]$

Satellite

DURING

Device

Idle $[1, \infty]$

Send2 $[5, 5]$

Evolution

Send1 $\in [30, \infty]$

- **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. `DEVICE.SEND2 ∈ [30, ∞]`)
**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 ∈ [30, ∞]`)

### Evolution

<table>
<thead>
<tr>
<th>Satellite</th>
<th>Hidden</th>
<th>Visible</th>
<th>Hidden</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Idle</td>
<td>Send1</td>
<td>Idle</td>
<td>Send2</td>
</tr>
</tbody>
</table>

Satellite and Device state transitions over time `t`.
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 ∈ [30, ∞)`)

![Diagram of timelines with states and transitions](Diagram.png)

Evolution

<table>
<thead>
<tr>
<th>Satellite</th>
<th>Hidden</th>
<th>Visible</th>
<th>Hidden</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device</th>
<th>Idle</th>
<th>Send1</th>
<th>Idle</th>
<th>Send2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timeline intervals:
- Satellite: Hidden [10, 12], Visible [10, 11]
- Device: Send1 [5, 5], Idle [1, ∞], Send2 [5, 5]

Timeline:
- Time `t`: 0, 10, 15, 21, 33, 35, 40

Timeline states:
- Satellite: Hidden, Visible
- Device: Idle, Send1, Send2
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 ∈ [30, ∞]`)

Evolution

<table>
<thead>
<tr>
<th>Satellite</th>
<th>Hidden</th>
<th>Visible</th>
<th>Hidden</th>
<th>Visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Idle</td>
<td>Send1</td>
<td>Idle</td>
<td>Send2</td>
</tr>
</tbody>
</table>

0  11  16  21  33  35  40  t
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 ∈ [30, ∞))

Evolution

Satellite

Device

0  10  16  20  32  35  40  

Hidden  Visible  Hidden  Visible

Idle  Send1  Idle  Send2
We annotate the domain values with **controllable** or **uncontrollable** flags for both starting and ending time.

- **Evolution**
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.
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

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.
Outline

1. Introduction
2. Timelines with Temporal Uncertainty
3. Strong Controllability Bounded-Horizon Encoding
4. Conclusion
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 within a given horizon.

Visible: $[10, 11]$

Hidden: $[10, 12]$

Send1: $[5, 5]$

DURING Idle: $[1, \infty]$

Satellite Device With horizon $H = 240$ 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.
Idea: we assume all durations positive and fix (an upper bound of) the maximal number of value changes for each generator within a given horizon.

Example

With horizon $H = 240$ 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 within a given horizon.

**Example**

With horizon $H = 240$ 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Problem</th>
<th>Monolithic</th>
<th>Incremental</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time(s)</td>
<td>Memory(Mb)</td>
</tr>
<tr>
<td>Sat</td>
<td>Satellite</td>
<td>6.87</td>
<td>111.5</td>
</tr>
<tr>
<td></td>
<td>TO</td>
<td>111.5</td>
<td>360.15</td>
</tr>
<tr>
<td></td>
<td>MO</td>
<td>1897.0</td>
<td>182.52</td>
</tr>
<tr>
<td>Sat</td>
<td>Machinery1</td>
<td>104.86</td>
<td>253.7</td>
</tr>
<tr>
<td></td>
<td>Meeting</td>
<td>23.12</td>
<td>630.8</td>
</tr>
<tr>
<td>Unsat</td>
<td>Satellite</td>
<td>7.17</td>
<td>126.2</td>
</tr>
<tr>
<td></td>
<td>Machinery2</td>
<td>113.53</td>
<td>284.4</td>
</tr>
<tr>
<td></td>
<td>Meeting</td>
<td>23.12</td>
<td>630.8</td>
</tr>
</tbody>
</table>
Outline

1. Introduction

2. Timelines with Temporal Uncertainty

3. Strong Controllability Bounded-Horizon Encoding

4. Conclusion
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!


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)
- $\text{start}(A)$ at 0
- $\text{start}(B)$ at 11
Schedules and Strategies Examples

Example

Dynamic Strategy (Dynamic Controllability)

- \textit{start}(A) \text{ at } 0
- \textit{start}(B) \text{ at } A_e

A ([7, 11])

B ([8, 11])
Example

Clairvoyant Strategy (Weak Controllability)

- \( \text{start}(A) \) at 0
- \( \text{start}(B) \) at \( A_e - 1 \)

\[ [7, 11], \quad B_e [8, 11], \quad B_s [8, 11] \]

\[ A_s [0, 20], \quad A_e [-1, \infty) \]
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 $\phi$, $\phi$ is satisfiable if there exists a model $\mu$ such that $\mu \models \phi$. 

Example $\phi \equiv (\forall x. (x > 0) \lor (y \geq x)) \land (z \geq y)$ is satisfiable in the theory of linear real arithmetic because $\mu = \{(y, 6), (z, 8)\}$ is a model that satisfies $\phi$. 

Theories Various theories can be used. In this work: LRA (Linear Real Arithmetic) QF-LRA (Quantifier-Free Linear Real Arithmetic)
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 $\phi$, $\phi$ is satisfiable if there exists a model $\mu$ such that $\mu \models \phi$.

**Example**

$\phi \equiv (\forall x.(x > 0) \lor (y \geq x)) \land (z \geq y)$

is satisfiable in the theory of linear real arithmetic because

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

is a model that satisfies $\phi$. 
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 $\phi$, $\phi$ is satisfiable if there exists a model $\mu$ such that $\mu \models \phi$.

**Example**

$\phi \equiv (\forall x. (x > 0) \lor (y \geq x)) \land (z \geq y)$

is satisfiable in the theory of linear real arithmetic because

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

is a model that satisfies $\phi$.

**Theories**

Various theories can be used.

In this work:

- $\mathcal{LRA}$ (Linear Real Arithmetic)
- $QF_{\mathcal{LRA}}$ (Quantifier-Free Linear Real Arithmetic)
Quantifier Elimination

Quantifier Elimination Definition

A theory $T$ has quantifier elimination if for every formula $\Phi$, there exists another formula $\Phi_{QF}$ 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 $\Phi$, there exists another formula $\Phi_{QF}$ 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).

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

Different techniques exists:
- Fourier-Motzkin
- Loos-Weisspfenning
- ...
Quantifier Elimination for \textit{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 \textit{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 \( \psi = \exists x_r . \land_{i=0}^{N} \sum_{k=1}^{M} a_{ik} x_k \leq b_i \) 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 \( A_h \vdash b_i - \sum_{k=1}^{r_i-1} a_{ik} x_k \), for \( h \in [1, H_A] \)
- \( x_r \leq B_h \), where \( B_h \vdash b_i - \sum_{k=1}^{r_i-1} a_{ik} x_k \), for \( h \in [1, H_B] \)
- Inequalities in which \( x_r \) has no role. Let \( \phi \) be the conjunction of those inequalities.

The system is equivalent to \((\max_{h=1}^{H_A} (A_h) \leq x_r \leq \min_{h=1}^{H_B} (B_h)) \land \phi \) and to \((\max_{h=1}^{H_A} (A_h) \leq \min_{h=1}^{H_B} (B_h)) \land \phi \)

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

\[
\psi \iff (\land_{i=0}^{H_A} \land_{j=0}^{H_B} A_i \leq B_j) \land \phi
\]
Fourier-Motzkin Example

Fourier Motzkin Example: Step 1

Let $\psi \equiv \forall z.((z \geq 4) \rightarrow ((x < z) \land (y < z)))$.
We convert all the quantifiers in existentials and we compute the DNF of the quantified part of the formula.

$\psi \iff \neg \exists z.((z \geq 4) \land \neg((x < z) \land (y < z)))$
$\psi \iff \neg \exists z.((z \geq 4) \land \neg(x < z) \lor \neg(y < z)))$
$\psi \iff \neg \exists z.((z \geq 4) \land \neg(x < z)) \lor ((z \geq 4) \land \neg(y < z)))$

Fourier Motzkin Example: Step 2

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

$((z \geq 4) \land (z \leq x)) \iff (4 \leq x)$
$((z \geq 4) \land (z \leq y)) \iff (4 \leq y)$

Then, we rebuild the formula:

$\psi \iff \neg((4 \leq x) \lor (4 \leq y))$
$\psi \iff ((x < 4) \land (y < 4))$
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)

\[ A_s, A_e, B_s, B_e \text{ are Time Points } (X_c) \]
\[ \rightarrow \text{ represents Free Constraints } (C_f) \]
Temporal Problems (with Temporal Uncertainty)

**Temporal Problems**

\[ A_s, A_e, B_s, B_e \text{ are Time Points (} X_c \text{)} \]

\[ \rightarrow \text{ represents Free Constraints (} C_f \text{)} \]

**Temporal Problems with Uncertainty**

\[ A_s, A_e, B_s \text{ are Controllable Time Points (} X_c \text{)} \]

\[ B_e \text{ is an Uncontrollable Time Point (} X_u \text{)} \]

\[ \rightarrow \text{ represents Free Constraints (} C_f \text{)} \]

\[ \ldots \rightarrow \text{ represents Contingent Constraints (} C_c \text{)} \]
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

\[ \frac{22}{9} \]
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
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
- \( \text{start}(A) \) at 0
- \( \text{start}(B) \) at 11

### Dynamic Strategy
- \( \text{start}(A) \) at 0
- \( \text{start}(B) \) 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**
- \( \text{start}(A) \) at 0
- \( \text{start}(B) \) at 11

**Dynamic Strategy**
- \( \text{start}(A) \) at 0
- \( \text{start}(B) \) at \( C \)

**Clairvoyant Strategy**
- \( \text{start}(A) \) at 0
- \( \text{start}(B) \) at \( C - 1 \)