Approximate Stream Reasoning with Incomplete State Information

Fourth Stream Reasoning Workshop, Linköping, Sweden

Daniel de Leng

Artificial Intelligence and Integrated Computer Systems
Department of Computer and Information Science
Linköping University, Sweden

LINKÖPING UNIVERSITY

Introduction

Introduction

Consider runtime verification for checking whether an agent is behaving in a safe manner.

Example (Safety)

"A robot in an unsafe state should return to a safe state within 10 seconds"

Motivation: Incomplete information!

Metric Temporal Logic

We use **Metric Temporal Logic** (MTL) as a language for describing temporal rules that must hold.

Definition (MTL syntax)

The syntax for MTL is as follows for atomic propositions $p \in \text{Prop}$, temporal interval $I \subseteq (0, \infty)$, and well-formed formulas (wffs) ϕ and ψ:

$$p \mid \neg\phi \mid \phi \vee \psi \mid \phi \, \mathcal{U}_I \, \psi$$

where $\Box_I$ and $\Diamond_I$ are syntactic sugar for 'always' and 'eventually'.

Progression-based Runtime Verification

Progression is an incremental **syntactic rewriting procedure** introduced by Bacchus and Kabanza (1996, 1998):

MTL Formula + Complete State + Delay ⇒ MTL Formula

$$\frac{\phi_0 = \Box(\neg p \rightarrow \Diamond_{[0,10]} p), s = \{\neg p\}, \Delta = 2}{\phi_1 = \Diamond_{[0,8]} p \land \Box(\neg p \rightarrow \Diamond_{[0,10]} p)}$$

Stream Reasoning with Incomplete Information

Problem: How to perform progression with **incomplete** states?

General idea: Apply model counting

Incomplete States and Streams

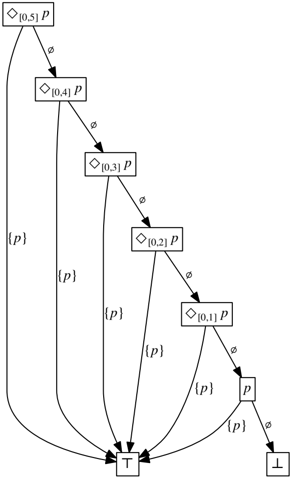
Important assumptions:

- We keep a constant delay value (Δ) and omit it from here on;
- An **incomplete state** $\widehat{s}$ is a disjunctive set of complete states;
- A (piecewise) **incomplete stream** $\widehat{\rho}$ is a sequence of incomplete states;
- We assume we have a probabilistic model of a stream denoted by a **state universe** $\mathbf{S}_n$ for every time-point n.

Progression Graphs

A **progression graph** encodes formulas and their progressions into a graph $G(\chi) = (\chi, V, E)$ such that

- vertices represent formulas;
- $\chi \in V$ represents the graph source formula; and
- labelled edges $(\phi, \psi, s) \in E$ iff $\mathtt{PROGRESS}(\phi, s) = \psi$.



Progression Graphs

Progression graphs $G_n(\chi) = (\chi, V, E, m_n)$ carry **probability mass**:

$$m_0(\chi) = 1.0 \text{ (Initialization)}$$

$$m_n(v) = \sum_{(v',v,s)\in E} \left(m_{n-1}(v')\ Pr[\mathbf{S}_n = s \mid \widehat{s}_n]\right)$$

Theorem (Soundness)

Given a progression graph $G_n(\chi)$ and a stream $\widehat{\rho}$:

$$\lim_{n\to\infty} m_n(\top) = Pr\left[\widehat{\rho}, t_0 \models \chi\right],$$

$$\lim_{n\to\infty} m_n(\bot) = Pr\left[\widehat{\rho}, t_0 \not\models \chi\right].$$

Progression Graph-Based Progression

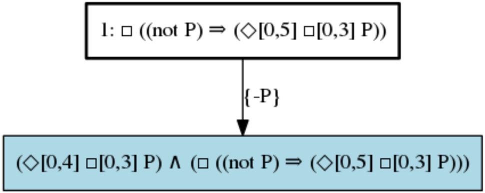
Example (Ship Stabilisation)

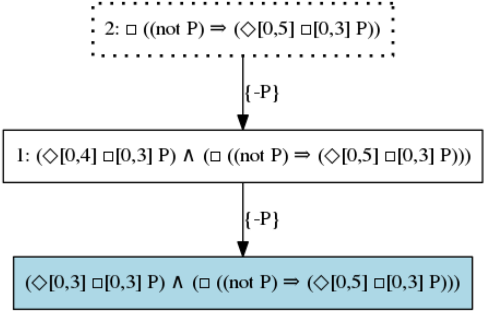
Suppose we have an autonomous ship with a landing deck. The ship attempts to stabilise itself according to the rule:

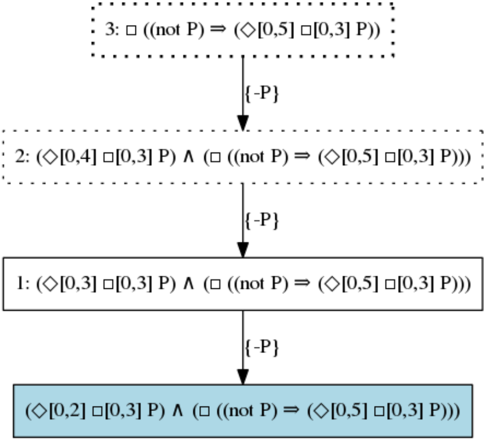
$$\Box(\neg p \rightarrow (\Diamond_{[0,5]}\Box_{[0,3]}p))$$

"Whenever the ship is unstable ($\neg p$), the ship will be stable (p) for a consecutive period of 3 minutes, within 5 minutes from having become unstable."

$$\Box\,((\text{not } P) \Rightarrow (\Diamond[0,5]\,\Box[0,3]\,P))$$


1: □ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
(◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))

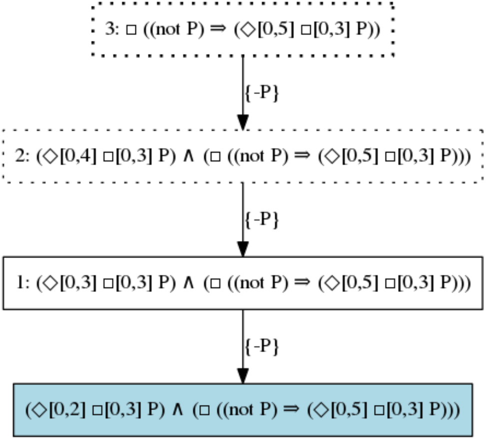

2: □ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
1: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
(◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))

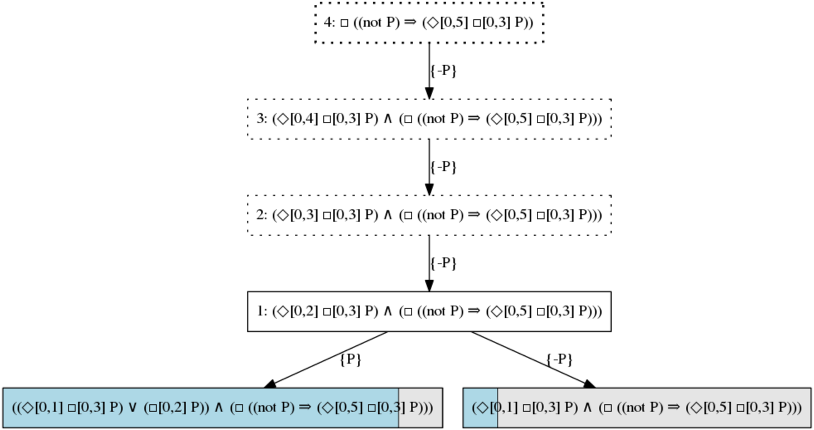

3: □ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
2: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
1: (◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
(◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))

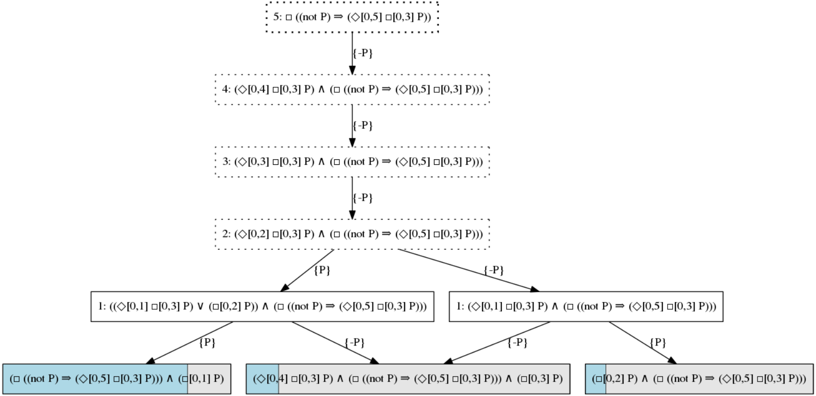
Incomplete Information

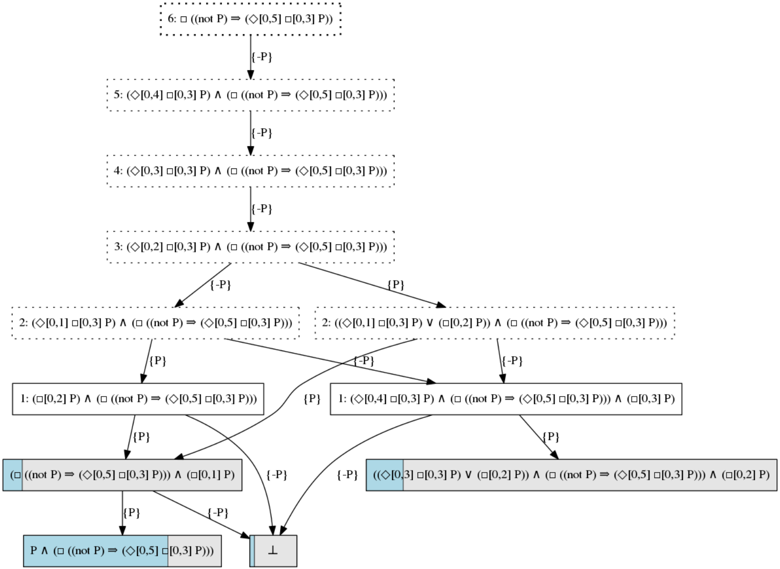
Example (Ship Stabilisation (Cont'd))

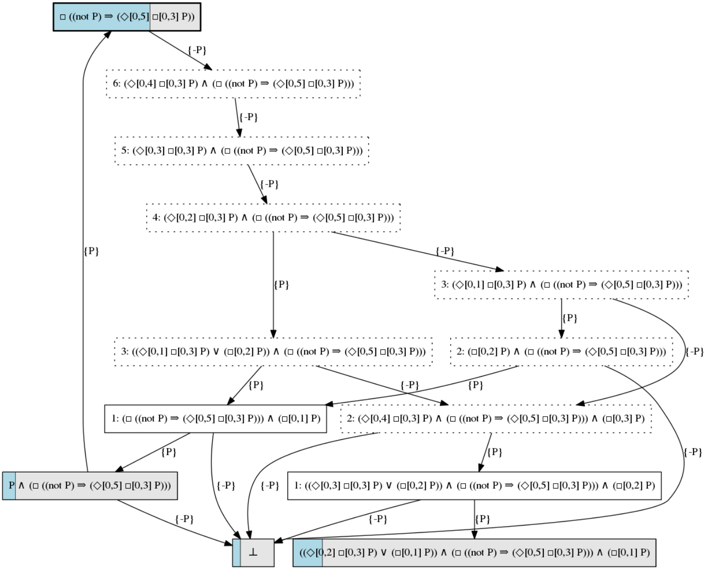
Suppose we are no longer able to measure unambiguously whether the ship is stable. Continue progression, and assume 90% stable, 10% unstable.

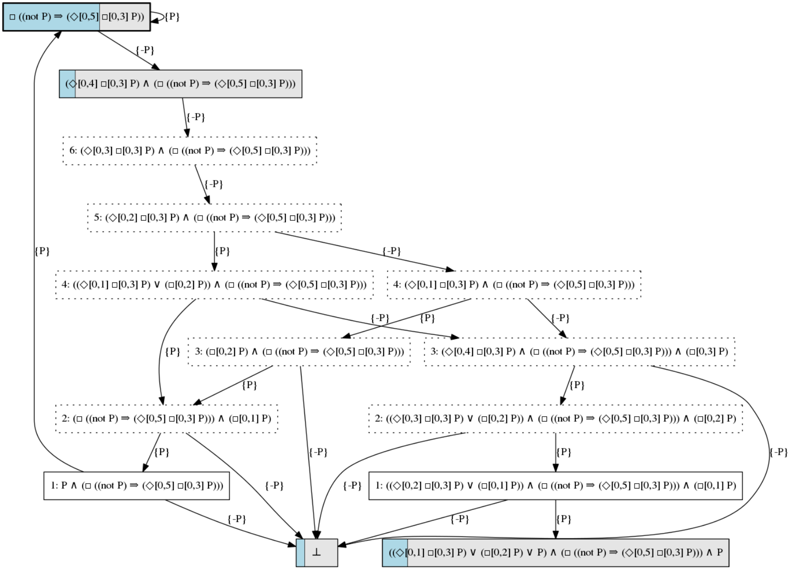

3: □ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
2: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
1: (◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
(◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))

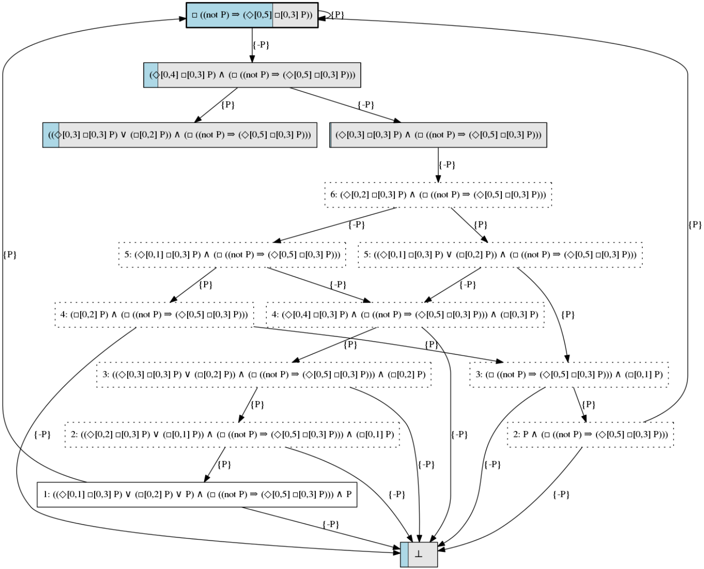

4: □ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
3: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
2: (◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
1: (◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
{-P}
((◇[0,1] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
(◇[0,1] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))

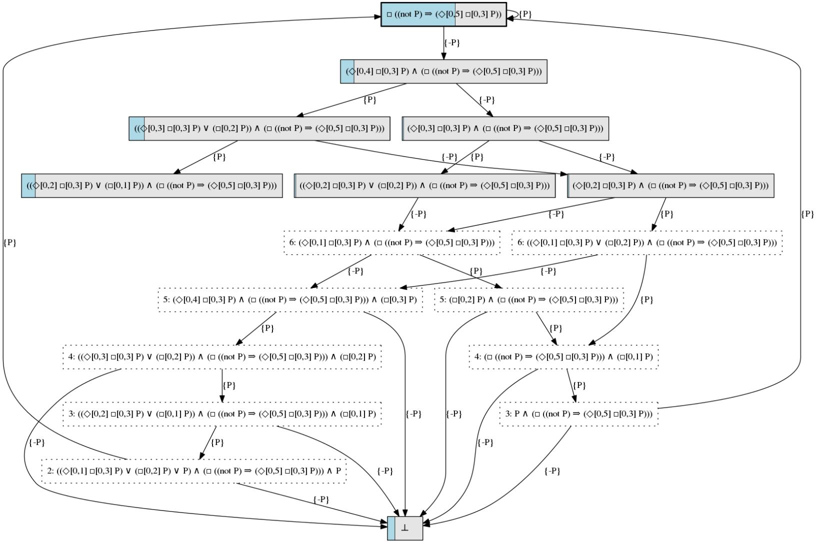

5: □ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
4: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
3: (◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
2: (◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
{-P}
1: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
1: (◇[0,1] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
{-P}
{-P}
{P}
(□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
(◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,3] P)
(□[0,2] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))


6: □ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
5: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
4: (◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
3: (◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
{P}
2: (◇[0,1] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
2: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
{-P}
{-P}
{P}
1: (□[0,2] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
1: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,3] P)
{P}
{P}
{-P}
{-P}
(□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
((◇[0,3] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,2] P)
{P}
{-P}
P ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
⊥


□ ((not P) ⇒ (◇[0,5] □[0,3] P))
{-P}
6: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
5: (◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
4: (◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
{P}
{-P}
3: (◇[0,1] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
3: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
2: (□[0,2] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
{P}
{-P}
{P}
1: (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
2: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,3] P)
{P}
{P}
{-P}
P ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
{-P}
1: ((◇[0,3] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,2] P)
{-P}
{-P}
{P}
⊥
((◇[0,2] □[0,3] P) ∨ (□[0,1] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)


□ ((not P) ⇒ (◇[0,5] □[0,3] P))
{P}
{-P}
(◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
6: (◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
5: (◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
4: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
4: (◇[0,1] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
3: (□[0,2] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
3: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,3] P)
2: (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
2: ((◇[0,3] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,2] P)
1: P ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
1: ((◇[0,2] □[0,3] P) ∨ (□[0,1] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
⊥
((◇[0,1] □[0,3] P) ∨ (□[0,2] P) ∨ P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ P


□ ((not P) ⇒ (◇[0,5] □[0,3] P))
{P}
{-P}
(◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
{-P}
((◇[0,3] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
(◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
6: (◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{-P}
{P}
5: (◇[0,1] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
5: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
{-P}
{-P}
{P}
4: (□[0,2] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
4: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,3] P)
{P}
{P}
3: ((◇[0,3] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,2] P)
3: (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
{P}
{P}
2: ((◇[0,2] □[0,3] P) ∨ (□[0,1] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
2: P ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
{P}
1: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P) ∨ P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ P
{P}
{P}
{-P}
{-P}
{-P}
{-P}
{-P}
{-P}
{-P}
{-P}
⊥


□ ((not P) ⇒ (◇[0,5] □[0,3] P))
(◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
((◇[0,3] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
(◇[0,3] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
((◇[0,2] □[0,3] P) ∨ (□[0,1] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
((◇[0,2] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
(◇[0,2] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
6: (◇[0,1] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
6: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
5: (◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,3] P)
5: (□[0,2] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
4: ((◇[0,3] □[0,3] P) ∨ (□[0,2] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,2] P)
4: (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
3: ((◇[0,2] □[0,3] P) ∨ (□[0,1] P)) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ (□[0,1] P)
3: P ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
2: ((◇[0,1] □[0,3] P) ∨ (□[0,2] P) ∨ P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P))) ∧ P
⊥
{P}
{-P}

Example: Ship Stabilisation

Example (Ship Stabilisation (Cont'd))

After 10 minutes, despite incomplete sensor readings, we know:

$$Pr[\widehat{\rho}, t_0 \not\models \Box(\neg p \rightarrow (\Diamond_{[0,5]}\Box_{[0,3]} p))] \geq 0.212680,$$

right now based on $m_{10}(\perp)$, *regardless* of future readings.

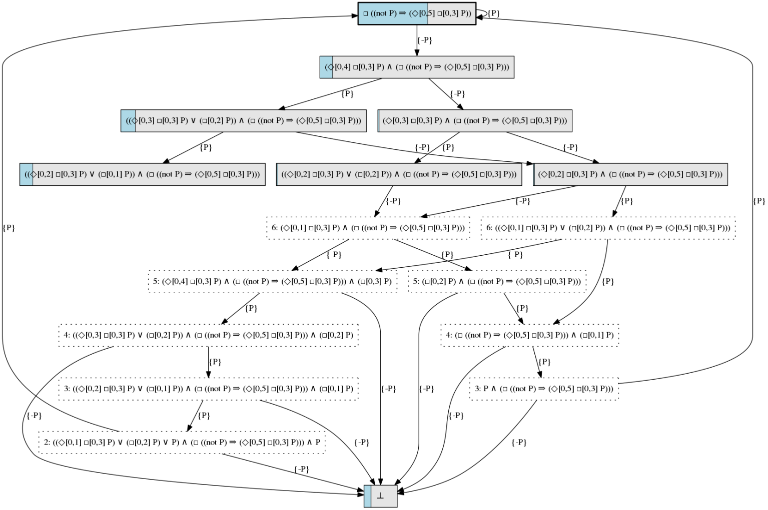
Approximate Progression

Approximate progression allows us to trade **precision** for **speed** and vice-versa:

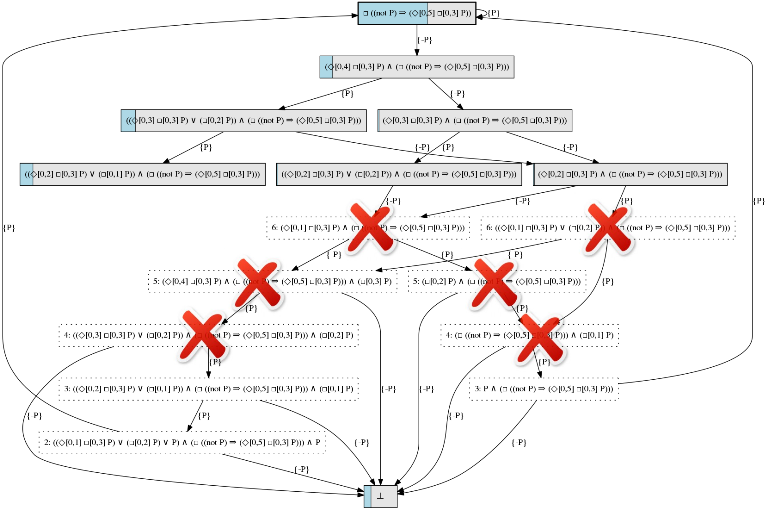
1. Institute a `MAX_AGE` for formulas;
2. Limit the size of the graph by setting a `MAX_NODES` bound.

We may drop nodes with probability mass, thereby **leaking** some probability mass over time.

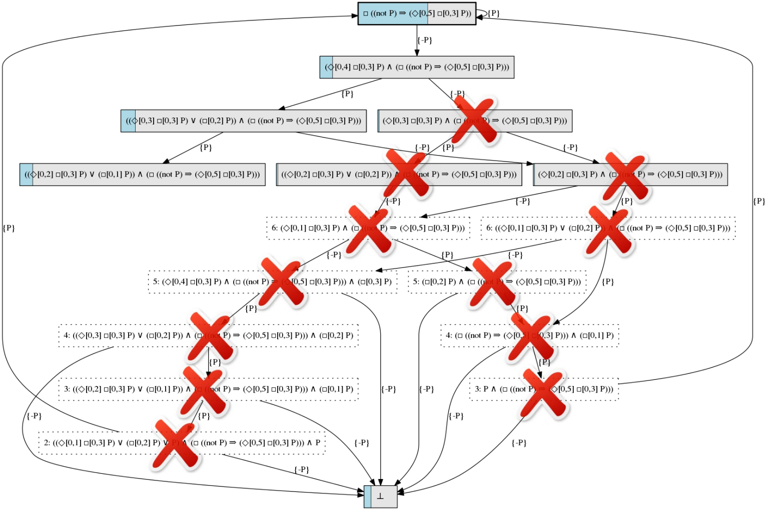
Methods to **reduce the graph size**: `MAX_AGE` and `MAX_NODES`.



Performance penalty: $\mathtt{MAX\_AGE} = 3$



Precision penalty: `MAX_NODES` = 5



Summary

Summary:

1. Classical progression assumes complete states;
2. We extended progression to handle incomplete states;
3. Progression graphs allow us to implicitly keep track of traces;
4. Approximation offers a trade-off between precision and speed.

Many interesting extensions possible!