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
Introduction

1. Introduction
2. Stream Reasoning with Incomplete Information
3. Progression Graph-Based Progression
4. Summary
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!
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) $\phi$ and $\psi$:

$$p \mid \neg\phi \mid \phi \lor \psi \mid \phi U_I \psi$$

where $\Box_I$ and $\Diamond_I$ are syntactic sugar for ‘always’ and ‘eventually’. 
Progression is an incremental **syntactic rewriting procedure** introduced by Bacchus and Kabanza (1996, 1998):

\[
\phi_0 = \Box(\neg p \rightarrow \Diamond [0,10] p), \ s = \{\neg p\} , \ \Delta = 2 \\
\phi_1 = \Diamond [0,8] p \land \Box(\neg p \rightarrow \Diamond [0,10] p)
\]
**Problem:** How to perform progression with **incomplete** states?

General idea: Apply model counting
Important assumptions:

- We keep a constant delay value ($\Delta$) and omit it from here on;
- An **incomplete state** $\hat{s}$ is a disjunctive set of complete states;
- A (piecewise) **incomplete stream** $\hat{\rho}$ is a sequence of incomplete states;
- We assume we have a probabilistic model of a stream denoted by a **state universe** $S_n$ for every time-point $n$. 
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 $\text{PROGRESS}(\phi, s) = \psi$. 

\[
\begin{align*}
\diamondsuit_{[0,5]} p \\
\diamondsuit_{[0,4]} p \\
\diamondsuit_{[0,3]} p \\
\diamondsuit_{[0,2]} p \\
\diamondsuit_{[0,1]} p \\
\top \\
\bot
\end{align*}
\]
Progression graphs $G_n(\chi) = (\chi, V, E, m_n)$ carry probability mass:

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

$$m_n(v) = \sum_{(v',v,s) \in E} (m_{n-1}(v') \cdot Pr[S_n = s | \hat{s}_n])$$

**Theorem (Soundness)**

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

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

$$\lim_{n \to \infty} m_n(\bot) = Pr[\hat{\rho}, t_0 \not\models \chi].$$
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.”
\( \square (\neg P) \Rightarrow (\lozenge [0,5] \square [0,3] P) \)
Introduction
Stream Reasoning with Incomplete Information
Progression Graph-Based Progression
Summary

Complete Information
Incomplete Information
Approximate Progression

Daniel de Leng
Linköping University

1: □ ((not P) ⇒ (◇[0,5] □[0,3] P))

{¬P}

(◇[0,4] □[0,3] P) ∧ (□ ((not P) ⇒ (◇[0,5] □[0,3] P)))
Introduction
Stream Reasoning with Incomplete Information
Progression Graph-Based Progression
Summary

Complete Information
Incomplete Information
Approximate Progression

Daniel de Leng
Linköping University 11/19
3: $\square((\neg P) \Rightarrow (\Diamond[0,5] \square[0,3] P))$

2: $(\Diamond[0,4] \square[0,3] P) \land (\square((\neg P) \Rightarrow (\Diamond[0,5] \square[0,3] P)))$

1: $(\Diamond[0,3] \square[0,3] P) \land (\square((\neg P) \Rightarrow (\Diamond[0,5] \square[0,3] P)))$

$(\Diamond[0,2] \square[0,3] P) \land (\square((\neg P) \Rightarrow (\Diamond[0,5] \square[0,3] P)))$
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)))
Stream Reasoning with Incomplete Information

Progression Graph-Based Progression

Summary

Complete Information

Incomplete Information

Approximate Progression

\[ 5: \Box \neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P) \]

\[ \neg P \]

\[ 4: (\Diamond [0.4] \Box [0.3] P) \land (\Box \neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P)) \]

\[ \neg P \]

\[ 3: (\Diamond [0.3] \Box [0.3] P) \land (\Box \neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P)) \]

\[ \neg P \]

\[ 2: (\Diamond [0.2] \Box [0.3] P) \land (\Box \neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P)) \]

\[ \{P\} \]

\[ \{\neg P\} \]

\[ 1: ((\Diamond [0.1] \Box [0.3] P) \lor (\Box [0.2] P)) \land (\Box \neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P)) \]

\[ \{P\} \]

\[ \{\neg P\} \]

1. \((\Box (\neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P))) \land (\Box [0.1] P)\)

2. \((\Diamond [0.4] \Box [0.3] P) \land (\Box (\neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P))) \land (\Box [0.3] P)\)

3. \((\Box [0.2] P) \land (\Box (\neg P \Rightarrow (\Diamond [0.5] \Box [0.3] P)))\)
Stream Reasoning with Incomplete Information

Introduction

Progression Graph-Based Progression

Summary

Complete Information

Incomplete Information

Approximate Progression

Daniel de Leng

Linköping University
Introduction
Stream Reasoning with Incomplete Information
Progression Graph-Based Progression
Summary

Complete Information
Incomplete Information
Approximate Progression

Daniel de Leng
Linköping University
13/19
Example: Ship Stabilisation

Example (Ship Stabilisation (Cont’d))

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

\[ Pr[\hat{\rho}, t_0 \not\models \square(\neg p \rightarrow (\lozenge[0,5] \square[0,3]p))] \geq 0.212680, \]

right now based on \( m_{10}(\bot) \), regardless of future readings.
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: \texttt{MAX\_AGE} and \texttt{MAX\_NODES}.
Performance penalty: $\text{MAX\_AGE} = 3$
**Precision** penalty: $\text{MAX}_\text{NODES} = 5$
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!