Partial-State Progression for Stream Reasoning with Metric Temporal Logic

Daniel de Leng, Fredrik Heintz
Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden
{daniel.de.leng, fredrik.heintz}@liu.se

Abstract
The formula progression procedure for Metric Temporal Logic (MTL), originally proposed by Bacchus and Kabanza, makes use of syntactic formula rewritings to incrementally evaluate MTL formulas against incrementally-available states. Progression however assumes complete state information, which can be problematic when not all state information is available or can be observed, such as in qualitative spatial reasoning tasks or in robot applications. Our main contribution is an extension of the progression procedure to handle partial state information. For each missing truth value, we efficiently consider all consistent hypotheses by branching progression for each such hypothesis. The resulting procedure is flexible, allowing a trade-off between faster but approximate and slower but precise partial-state progression.

Motivation
Metric Temporal Logic (MTL) by (Koymans 1990) extends Linear Temporal Logic (LTL) (Emerson 1990) by adding metric intervals for the temporal operators. The progression algorithm for MTL (Bacchus and Kabanza 1996; 1998) was developed to allow for the incremental evaluation of MTL formulas. This makes it possible to perform incremental reasoning over incremental information, called stream reasoning. Progression-based stream reasoning has for example previously been used on-board unmanned aerial vehicles (UAVs) for tasks such as planning and execution monitoring (Heintz and Doherty 2004; Kvarnström, Heintz, and Doherty 2008).

Progression works by incrementally reading states from a state sequence and computing a new formula that incorporates this state information using syntactic rewritings. If the new formula holds over the unseen remainder of the state sequence, then the original formula is guaranteed to hold over the complete state sequence. Consequently the evaluation of an MTL formula through progression is linear in the size of the formula, but the formula may grow exponentially due to the rewritings. Furthermore, once a formula is determined to be true or false, the answer can be returned due to monotonicity. One key assumption for progression is that the states received are complete, i.e. all propositions have a truth value assigned to them. This assumption is however unreasonable in applications in which acquiring such a snapshot is not feasible, e.g. robot applications relying on sensor data.

Partial-State Progression
The classical progression procedure, denoted by PROGRESS, corresponds to the progression of a stream of complete states. These full states have no uncertainty about the truth values of their contained propositions. The progression procedure for every time-point takes a wff \( \phi \) together with a (complete) state \( s \in 2^{\text{Prop}} \) and a time delay \( \Delta \in \mathbb{N} \), and produces a resulting formula \( \phi' \) which may be a verdict \( \top \) or \( \bot \). Since we only have to consider a single state every time we read a state, progression can be applied to that state, yielding a progressed formula \( \phi' \) that acts as the input formula for the next state. One shortcoming of classical progression is the requirement of complete rather than partial states. Partial states are states for which some or all propositions have an unknown truth assignment. We are able to model partial states, denoted by \( \hat{s} \), by representing them in DNF, i.e. a partial state \( \hat{s} \) is represented by a disjunctive set of complete states \( \{s_1, \ldots, s_n\} \) representing all possible complete interpretations of \( \hat{s} \). For example, if for a state \( \hat{s} \) we know \( p \) is true but we do not know the truth value of \( q \), then \( \hat{s} = \{\{p\}, \{p, q\}\} \) in the absence of additional background theories.

We propose partial-state progression using progression graphs, which are composed of formulas connected by directed edges labelled with sets of states. Each formula \( \phi \) has an outgoing edge—with a label containing complete states \( s \)—to a destination formula \( \phi' \) iff \( \text{PROGRESS}(\phi, s, \Delta) = \phi' \) for each of the complete states \( s \). Each formula additionally has a probability mass and a time-to-live (ttl) associated with it. The probability mass represents the ratio of traces that have currently reached an associated formula. When progressing a new formula \( \phi \), all of the probability mass resides in \( \phi \), denoted by \( m(\phi) = 1 \). While structurally similar to deterministic timed automata (DTA), progression graphs instead are used to push probability mass between nodes and consequently lack the notion of clocks or accepting states.

Algorithm 1 shows the leaky multi-state progression procedure \texttt{MP-LEAKY}, which takes a progression graph and a partial states, and yields an updated progression graph. Repeated application results in probability mass get-

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
Algorithm 1: Approximate Partial-State Progression

1 function MP-LEAKY (G, s):
2     V' ← V
3     foreach ψ ∈ V' do
4         if m(ψ) > 0 then
5             if ¬expanded(ψ) then
6                 foreach s ∈ 2^Prop do
7                     ψ' ← PROGRESS(ψ, s, ∆)
8                     if ψ' ∉ V' then
9                         V ← V ∪ {ψ'}
10                        expanded(ψ') ← false
11                 end
12             end
13             E ← E ∪ {(ψ, ψ', s)}
14             expanded(ψ) ← true
15         end
16     foreach (ψ, ψ', s) ∈ E do
17         m'(ψ) ← m'(ψ') + m(ψ)
18     end
19
20     Decrease ttl and remove expired formulas with ttl < 0.
21     Remove formulas by mass while |V| > MAX_NODES.
22     m ← m'.
23     return G

Figures 1 and 2 show the leaked probability mass at termination (left), and time to termination ±2σ (right).

In conclusion, the proposed partial-state progression procedure provides a trade-off between faster but approximate, and slower but precise partial-state progression. The proposed procedure could for example be useful in qualitative spatio-temporal stream reasoning (Heintz and de Leng 2014) to deal with the intrinsic uncertainty associated with qualitative spatial relations.

Acknowledgments

This work is partially supported by a grant from the National Graduate School in Computer Science, Sweden (CUGS).

References


