Proceedings of the Sixteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2018)

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 rewriting. 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 ϕ together with a (complete) state $s \in 2^{\mathsf{Prop}}$ and a time delay $\Delta \in \mathbb{N}$, and produces a resulting formula ϕ' 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 ϕ' 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 ϕ has an outgoing edge—with a label containing complete states s—to a destination formula ϕ' iff $\mathtt{PROGRESS}(\phi, s, \Delta) = \phi'$ 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 ϕ, all of the probability mass resides in ϕ, 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 MP−LEAKY, which takes a progression graph and a partial states, and yields an updated progression graph. Repeated application results in probability mass get-

Algorithm 1: Approximate Partial-State Progression

function MP-LEAKY $(G, \hat{s})$:
$V' \leftarrow V$
foreach $\psi \in V'$ **do**
  if $m(\psi) > 0$ **then**
    if $\neg expanded(\psi)$ **then**
      foreach $s \in 2^{\mathsf{Prop}}$ **do**
        $\phi' \leftarrow \mathtt{PROGRESS}(\psi, s, \Delta)$
        if $\psi' \notin V'$ **then**
          $V \leftarrow V \cup \{\psi'\}$
          $expanded(\psi') \leftarrow false$
        end
        $E \leftarrow E \cup \{(\psi, \psi', s)\}$
        $expanded(\psi) \leftarrow true$
      end
    end
    foreach $(\psi, \psi', s) \in E$ **do**
      $m'(\psi') \leftarrow m'(\psi') + \frac{m(\psi)}{|\hat{s}|}$
      $ttl(\psi') \leftarrow \mathtt{MAX\_TTL}$
    end
  end
end
Decrease ttl and remove expired formulas with $ttl < 0$.
Remove formulas by mass while $|V| > \mathtt{MAX\_NODES}$.
$m \leftarrow m'$
return G

ting stuck in verdict nodes $\top$ and/or $\bot$, which are fixpoints of PROGRESS. The MAX_TTL constant allows us to 'drop' formulas if they do not receive any probability mass for MAX_TTL time-units, without affecting precision. When MAX_NODES is set sufficiently high, the procedure is guaranteed to be precise. In the converse case, some formulas carrying probability mass are removed until MAX_NODES is reached, removing formulas with the least probability mass. The 'leaked' probability mass corresponds to unlikely hypothetical traces that were chosen to not be pursued, and allows us to quantify the precision of the progression graph in terms of how much mass was leaked.

We performed initial empirical evaluations[1] using a fourth-generation Intel Xeon E5-1650 CPU (6 cores, 12 threads) with 50GiB of RAM allocated to the JVM. Figure 1 shows an analysis of probability mass leakage for an example MTL formula $\Box\left(\neg p \rightarrow \left(\Diamond_{[0,100]}\left(\Box_{[0,10]} p\right)\right)\right)$ with $\Delta = 1$ and $\mathtt{MAX\_TTL} = 1$. On the left-hand side (corresponding to the blue dashed line), we see the leaked probability mass at termination, which occurs when 99% of probability mass has found its way into verdict nodes for 'false' or was leaked (corresponding to 'unknown'). As the probability of an 'unknown' verdict decreases, the probability of a $\bot$ verdict—here the inverse; not shown explicitly—increases. When we decrease the value for MAX_NODES, the 'unknown' verdict dominates the 'false' verdict, and vice-versa. On the right-hand side of Figure 1 (the red line) we see the time to termination and the two standard deviations (the red shade). To better illustrate the velocity of the state stream, consider that for $\mathtt{MAX\_NODES} = 175$ a total of 222,599 states were fed to MP-LEAKY to reach the termination criterion within 89.174s. This corresponds to the ability to handle a sample rate of a bit under 2,500Hz.

[1]The jprogress implementation is available at https://github.com/dnleng/jprogress.

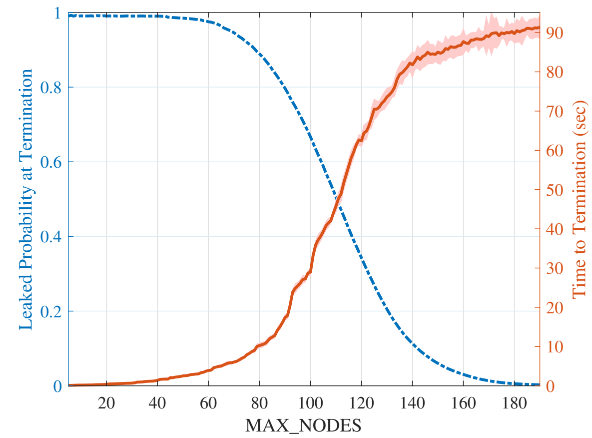

Figure 1: Leaked probability mass at termination (left), and time to termination $\pm 2\sigma$ (right).

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

Bacchus, F., and Kabanza, F. 1996. Planning for temporally extended goals. In *Proc. AAAI*, 1215–1222.

Bacchus, F., and Kabanza, F. 1998. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence* 22(1-2):5–27.

Emerson, E. A. 1990. Temporal and modal logic. In *Formal Models and Semantics*. Elsevier. 995–1072.

Heintz, F., and de Leng, D. 2014. Spatio-temporal stream reasoning with incomplete spatial information. In *Proc. ECAI*, 429–434.

Heintz, F., and Doherty, P. 2004. DyKnow: An approach to middleware for knowledge processing. *Journal of Intelligent and Fuzzy Systems* 15(1):3–13.

Koymans, R. 1990. Specifying real-time properties with Metric Temporal Logic. *Real-Time Systems* 2(4):255–299.

Kvarnström, J.; Heintz, F.; and Doherty, P. 2008. A temporal logic-based planning and execution monitoring system. In *Proc. ICAPS*, 198–205.