Incremental Dynamic Controllability:  
A Sound Verification Algorithm  
Mikael Nilsson, Jonas Kvarnström, Patrick Doherty

**Summary**

Simple Temporal Networks with Uncertainty (STNU) allow the representation of constraints on actions whose durations may vary within given bounds. An STNU is executable if it is dynamically controllable. We revisit FastIDC, a current state of the art incremental dynamic controllability verification algorithm, and:

- Show it is unsound
- Provide an efficient solution to restore soundness

**STNUs and Dynamic Controllability**

In the STNU formalism temporal networks are used to represent events (nodes) and constraints (edges) between them. As an example, suppose I want to prepare food as a surprise for my wife after she returns from shopping.

**A dynamic execution strategy** is a strategy for assigning times to controllable events during execution, given that we observe uncontrollable events as they happen. The strategy must ensure that all requirement constraints will be respected regardless of the outcomes of uncontrollable events. An STNU is dynamically controllable (DC) if there exists a dynamic execution strategy for it.

In the example above a dynamic execution strategy is to start cooking 4 minutes after observing the start car event.

**Incremental DC Verification**

The FastIDC algorithm repeatedly applies derivation rules to expose implicit constraints in the STNU.

**Flaws in FastIDC**

- The authors confounded derivation rules D8 and D9
- The algorithm is unsound even with a correct interpretation of D8 and D9.

**Problem:**

Plans that are believed to be dynamically controllable may fail at execution time.

**Example**

1. In this example we build an STNU incrementally with FastIDC. In the STNU U is an uncontrollable event that will be observed 5 to 50 time units after A is executed. The rest of the edges represent requirement constraints.

   ![Dynamic Execution Example](image)

   However by inspection we find the U4 constraint, requiring U to be executed at least 50 time units after A, This STNU is not DC since this cannot be guaranteed!

**An Efficient Solution**

We proved that much of the work done by FastIDC could be used to detect and correctly classify the previously misclassified instances.

**Lemma**

Consider all requirement paths of a given weight between two nodes in a distance graph that was constructed incrementally by FastIDC. The shortest of these paths, in terms of the number of edges, must have one of the following forms:

1. It contains only positive edges.
2. It consists of at least one negative edge followed by zero or more positive edges.

**Theorem**

If a dynamically controllable STNU constructed using FastIDC becomes non-DC with the addition or tightening of an edge the distance graph contains a cycle consisting of only negative edges.

We detect such cycles efficiently using an incremental topological ordering algorithm which does not increase the run-time complexity of FastIDC.

**References**