Optimal Manufacturing Controller Synthesis Using Situation Calculus

Omar Adalat Daniele Scrimieri Savas Konur
University of Bradford, UK

Introduction

- **Controller synthesis in manufacturing systems** has the goal of synthesizing a *process controller* using:
  - a high-level target **process recipe**
  - lower-level behaviour formalised of **production resources** (robots, tools, sensors/probes, and machinery).
- **Situation Calculus**: a many-sorted second order logic for reasoning about actions with dynamic predicates, objects, and actions. All changes to the world are a result of named actions, with specified action preconditions and effect axioms. Facility actions are *compound*, with each production resource taking an atomic action (which may include no-operation/idling). Compound actions can have different rules applied contrasted to individually performing the constituent actions.

  ConGolog: a high-level *programming language* for the situation calculus, which allows introducing *non-determinism* & *true concurrency* (parallelism).
  Characteristic graphs: allow modelling ConGolog program execution states and transitions in a succinct manner (actions + conditions + variables).
- **Production resources** (robots and all equipment) are modelled with logical action theories + high-level ConGolog programs. Actions involve idling, preparatory actions (e.g. equipping bits, synchronized transferring parts between resources) and manufacturing operations (e.g. drilling a part, gluing parts, etc).
- **Process recipes** are modelled simply with a high-level ConGolog program, that only addresses high-level manufacturing behaviours (completely resource-agnostic, ignoring lower-level operations such as part transfers).

Objectives of controller synthesis

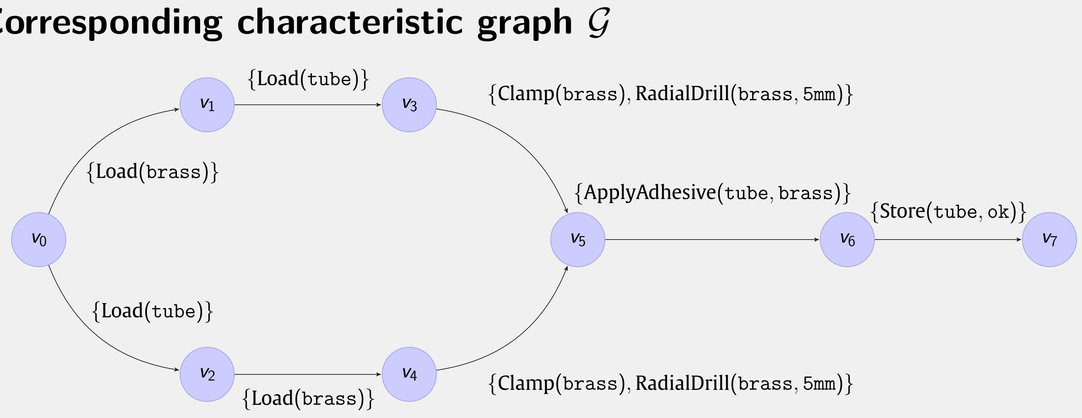
- **Finding optimal controllers** i.e. a controller which minimises the number of true concurrent and individual actions, or cost ascribed therewith.
- **Minimizing execution time of synthesis** our methods involve progression (forward search) which are amenable to heuristics.

Illustrative process recipe

ConGolog recipe program

$$\begin{gathered}\text{Load}(\texttt{brass}) \;||\; \text{Load}(\texttt{tube});\\ \text{Clamp}(\texttt{brass}) \;|||\; \text{RadialDrill}(\texttt{brass}, 5);\\ \text{ApplyAdhesive}(\texttt{tube}, \texttt{brass});\; \text{Store}(\texttt{brass}, \texttt{ok})\end{gathered}$$

Corresponding characteristic graph $\mathcal{G}$



Description

- In an interleaved fashion, load a brass part and a tube part. This can be done simultaneously, or potentially either after the other (weak concurrency).
- The brass part must be clamped and drilled - simultaneously/joint execution (true concurrency).
- Then, in sequence, apply adhesive and store the product with code ok.

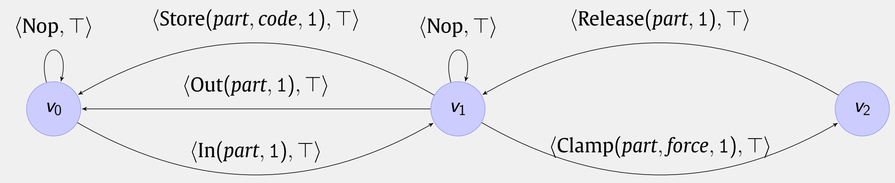
Illustrative production resource

A production resource's ConGolog program

loop:

$$\begin{gathered}\text{Nop}^* \mid (\text{In}(\textit{part}, 1); (\text{Nop} \mid (\text{Clamp}(\textit{part}, \textit{force}, 1); \text{Release}(\textit{part}, 1)))^*;\\ (\text{Out}(\textit{part}, 1) \mid \text{Store}(\textit{part}, \textit{code}, 1))\end{gathered}$$

Corresponding characteristic graph $\mathcal{G}$



Logical basic action theory $\mathcal{D}$

$$\begin{gathered}\text{Poss}(\text{Load}(\textit{part}, i, s) \equiv \textit{Part}(\textit{part}, s) \wedge \textit{on\_site}(\textit{part}, s)\\ \textit{on\_site}(\textit{part}, \text{do}(\mathbf{a}, s)) \equiv \textit{on\_site}(\textit{part}, s) \wedge \neg\exists\, i.\ \text{Load}(\textit{part}, i) \in \mathbf{a} \wedge \neg\exists\, \textit{code}, i.\ \text{Store}(\textit{part}, \textit{code}, i) \in \mathbf{a}\\ \textit{part}(\textit{part}, \text{do}(\mathbf{a}, s)) \equiv \textit{part}(\textit{part}, s) \wedge \neg\exists\, \textit{code}, i.\ \text{Store}(\textit{part}, \textit{code}, i) \in \mathbf{a} \wedge \exists\, \textit{other}, i.\ \text{ApplyAdhesive}(\textit{other}, \textit{part}, i) \in \mathbf{a}\end{gathered}$$

where Poss represents a precondition axiom and the other two formulae represent successor state axioms (changes the fluent's truth valuation based on the performed action from the binary function do).

Description

- The *loop*: construct means we are dealing with an **infinite** execution flow.
- The resource may indefinitely idle from the Nop* construct - another source of infinite execution.
- It may also take a *part* in from another resource, then potentially indefinitely follow an execution sequence of Nop or clamping with a particular force and a successive release in the next timestep. Proceeding this, it may transfer the *part* out to another resource, or it may store the *part* with a particular *code*.

Synthesis strategy & Future work

- We use a **progression** based search (forward search), imposing that all potential recipes are finite processes. We use a *fairness* assumption for strong cyclical planning, enforcing limits on infinite executions of resources.
- Heuristics can be used in a progression-based search by ascribing costs to atomic & compound actions and checking progress completion of controller candidates, for example A* and Greedy Best First Search.

Future work

- Can we **improve** non-optimal plans to optimality?
- Enhanced representations: **non-Markovian actions**, **explicit time**, **noisy and fallible sensing**, etc.

References

- De Giacomo, G., Felli, P., Logan, B., Patrizi, F. and Sardina, S., 2022. *Situation calculus for controller synthesis in manufacturing systems with first-order state representation*. Artificial Intelligence, 302, p.103598.
- Sardina, S. and De Giacomo, G., 2009, June. *Composition of ConGolog programs*. In Twenty-First International Joint Conference on Artificial Intelligence.

This work was supported by the SURE Research Projects Fund and Research Development Fund of the University of Bradford and Innovate UK grant no: 10028947 (Made Smarter Innovation: Sustainable Smart Factory).