CHALMERS
UNIVERSITY OF TECHNOLOGY

Identification of Minimally Restrictive Assembly Sequences Using Supervisory Control Theory

Citation for the original published paper (version of record):
Vinetti, M., Fabian, M. (2025). Identification of Minimally Restrictive Assembly Sequences Using Supervisory Control Theory. International Conference on Industrial Cyber Physical Systems Icps. http://dx.doi.org/10.1109/ICPS65515.2025.11087917

Identification of Minimally Restrictive Assembly Sequences using Supervisory Control Theory

Martina Vinetti
Department of Electrical Engineering
Chalmers University of Technology
Göteborg, Sweden
vinetti@chalmers.se

Martin Fabian
Department of Electrical Engineering
Chalmers University of Technology
Göteborg, Sweden
fabian@chalmers.se

***Abstract*—Modern assembly processes require flexibility and adaptability to handle increasing product variety and customization. Traditional assembly planning methods often prioritize finding an optimal assembly sequence, overlooking the requirements of contemporary manufacturing. This work uses *Supervisory Control Theory* to systematically generate *all* feasible assembly sequences while ensuring compliance with precedence and process constraints. By synthesizing a controllable, non-blocking, and minimally restrictive supervisor, the proposed method guarantees that only valid sequences are allowed, balancing flexibility and constraint enforcement. The obtained sequences can serve as a basis for further optimization or exception management, improving responsiveness to disruptions.**

I. Introduction

In recent years, assembly processes have encountered increasing challenges due to evolving market demands and technological advancements [1]. The explosion in product variants [2] and the growing emphasis on customization [3] have added complexity to assembly lines, requiring greater flexibility and adaptability while maintaining efficiency and quality.

These challenges are especially critical in manual assembly, where workers must manage increasing product variations and adapt task sequences to specifications, disruptions, or resource availability. This underscores the need for flexible and automated support in assembly sequence planning to ensure efficiency while reducing cognitive overload.

Traditional assembly planning methods often emphasize the search for a single optimal sequence, a challenge known as Assembly Sequence Planning (ASP) [4], but may struggle to meet the demands of modern manufacturing. These approaches prioritize optimization metrics, such as cost, time, and efficiency [5], [6], over the flexibility and adaptability needed to handle dynamic and uncertain production environments.

This work addresses the challenge of generating *all* feasible assembly sequences that meet specified constraints.

This is achieved by using the Supervisory Control Theory (SCT) which, in contrast to conventional methods that often rely on specific representations like interference matrices [5] and precedence graphs tailored to particular constraints [4], provides a unified and flexible modeling framework.

Assembly tasks and the constraints between them are represented as finite automata (FAs), enabling the seamless incorporation of both static and dynamic constraints while maintaining a comprehensive perspective of the assembly process. A key innovation of the proposed method is its ability to generalize and model complex assembly processes beyond simple precedence relationships. Additionally, this method avoids blocking configurations, ensuring that every feasible sequence leads to completion without constraint violations. SCT's formal synthesis retains only transitions leading to reachable final states, inherently guaranteeing non-blocking behavior.

The minimally restrictive nature of the synthesized supervisor, an inherent property of SCT, ensures that assembly tasks are restricted only when necessary, balancing feasibility and flexibility. This guarantees that any sequence obtained allows the completion of the entire assembly process while satisfying all constraints, addressing the need for adaptive and robust assembly planning in complex and dynamic manufacturing environments.

The generation of all feasible assembly sequences offers significant advantages. It confines the search space for further optimization, making it easier to identify the most suitable sequence according to specific performance criteria. Also, it provides a foundation for exception management systems, enabling the identification of alternative sequences when disruptions occur, such as missing components or unavailable tools. Moreover, in manual assembly, by giving operators various feasible sequences, adaptability on the shop floor is improved, enabling workers to select the sequence that best suits their current context. This aligns with the principles of Industry 5.0 [7], fostering a human-centric manufacturing environment where collaboration between advanced technologies and human operators enhances flexibility, personalization, and overall efficiency.

II. Supervisory Control Theory

A discrete event system can be modeled using finite automata (FAs) [8]. An FA is defined as a 5-tuple $\langle Q, \Sigma, \delta, q_i, Q_m \rangle$, where Q represents the finite set of states,

This work was supported by the EUREKA ITEA4 ArtWork project (2023-00970), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Σ is the finite set of events, $\delta : Q \times \Sigma \to Q$ is the partial transition function, $q_i \in Q$ is the initial state, and Q_m is the set of marked states.

In Supervisory Control Theory (SCT) [9], a system's behavior is modeled by an automaton known as the plant P, which represents all possible behaviors of the system. To ensure that the system meets specific desired outcomes, the plant is restricted by the specification K, another automaton that defines the desired behavior.

When multiple FAs interact, their combined behavior can be represented by the *synchronous composition* operator. This operator models the concurrent execution of two FAs by ensuring that shared events occur simultaneously in both systems while independent events evolve freely in their respective FA. Formally, given two FAs $A = (Q_A, \Sigma_A, \delta_A, q_i^A, Q_m^A)$ and $B = (Q_B, \Sigma_B, \delta_B, q_i^B, Q_m^B)$, their synchronous composition is defined as the FA:

$$A \parallel B = \langle Q_A \times Q_B, \Sigma_A \cup \Sigma_B, \delta, (q_i^A, q_i^B), Q_m^A \times Q_m^B \rangle$$

where the transition function δ is given by:

$$\delta((q^A, q^B), \sigma) = \begin{cases} (\delta_A(q^A, \sigma), \delta_B(q^B, \sigma)) & \text{if } \sigma \in \Sigma_A \cap \Sigma_B \\ (\delta_A(q^A, \sigma), q^B) & \text{if } \sigma \in \Sigma_A \setminus \Sigma_B \\ (q^A, \delta_B(q^B, \sigma)) & \text{if } \sigma \in \Sigma_B \setminus \Sigma_A \end{cases}$$

From the composition $P||K$, a supervisor [9] S can be synthesized such that the controlled system $P||S$ fulfills $P||K$. Some events are *uncontrollable* and thus not subject to influence by S, which must be taken into account by the synthesis. Also, S must enforce the specification while guaranteeing that a marked state is always accessible from any reachable state, thus ensuring *non-blocking* behavior. Lastly, S should be *minimally restrictive*, meaning that it does not impose more control than absolutely necessary to fulfill K.

The software tool SUPREMICA [10] facilitates the synthesis of supervisors and provides formal verification to ensure that these supervisors satisfy the requirements of controllability, non-blocking, and minimal restrictiveness. SUPREMICA has been applied in various domains, including industrial research projects related to robotics and manufacturing systems [11].

III. METHOD

Identifying feasible assembly sequences requires a precise understanding of the tasks involved in the assembly process and the constraints governing their execution. This study considers a scenario where a set of tasks and their constraints are given, without addressing coordination between multiple agents (robots or workers). However, the approach can be easily extended to multi-agent settings by incorporating additional constraints on inter-agent task dependencies. In the considered case, constraints are categorized as either *static* or *dynamic*, both imposing limitations on the possible assembly sequences.

Static constraints, such as precedence constraints, impose fixed limitations on the order of execution. These arise primarily from product design requirements, ensuring that parts are assembled in a way that satisfies technical and functional specifications. Additionally, some precedence constraints are dictated by process dependencies, which are not derivable solely from the product's geometry but stem from operational requirements or interconnections with other components.

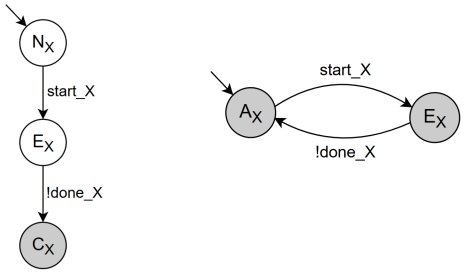
Beyond static constraints, real-time conditions may impose dynamic constraints. Examples include tool/resource dependencies, where one task provides essential tools or resources for another; collision-free path dependencies, which ensure that assembly movements do not interfere with each other; and stability dependencies, where the structural stability of a subassembly depends on another ongoing operation.

To systematically represent both tasks and constraints, a formal modeling approach based on FAs in SUPREMICA is adopted. The following section details how these elements are encoded within this framework.

A. Modeling

Each task in the assembly process is modeled as an individual finite automaton. Since, in a real system, only the initiation of a task can be controlled while its completion depends on uncontrollable factors, each task automaton has a controllable event representing the task's start and an uncontrollable event indicating its completion. As a result, tasks can be initiated or completed while others are executing, enabling the representation of process dependencies such as tool availability, collision-free path generation, or subassembly stability.

Two types of tasks are considered, see Figure 1. Tasks that are executed only once during the assembly process are modeled as three-state automata, where the execution follows a finite sequence of transitions, ensuring a strict progression from initiation to completion. The final state is marked, as it represents the desired completion of the task. Conversely, tasks that require multiple executions, such as screwing operations or visual inspections, are represented using loop automata, with both states marked, allowing their execution to be repeated as needed within the process.



(a) Single-execution task modeling (b) Repetitive task modeling

Fig. 1: FAs of a generic task X: (a) single-execution task with states *Non-completed*, *Executing*, and *Completed*; (b) repetitive task with states *Available* and *Executing*.

Both tasks and constraints can be modeled in multiple ways, based on states and events.

B. Supervisor Synthesis

Once the plant (the tasks) and the specifications (the constraints) have been modeled and synchronized, a supervi-

sor [9] is synthesized in SUPREMICA. Supervisor synthesis is an iterative procedure that eliminates undesirable states, specifically those from which no marked state is reachable or where uncontrollable events could lead to such states. As is known [9], this procedure converges to a fixpoint, yielding the least restrictive supervisor that ensures both controllability and nonblocking behavior. The supervisor guarantees that all allowed task sequences comply with the given constraints while maximizing system flexibility and ensuring that some marked state is always reachable.

IV. CASE STUDY

A manageable case study has been selected, which allows for the manual verification of the expected behavior. It regards an assembly process consisting of five main tasks: *A*, *B*, *C*, *D*, and *E*. Each task must be executed exactly once for the assembly process to be considered complete. The precedence constraints among these tasks are represented by the directed graph (*digraph*) in Figure 2, where nodes correspond to tasks and directed edges indicate that a task must be done before the next can start.

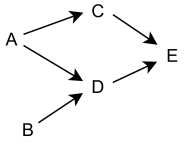


Fig. 2: Digraph of precedence constraints among tasks.

In SUPREMICA, these tasks are represented as illustrated in Figure 1 (a). In addition to static precedence constraints, dynamic constraints are also introduced.

Two dynamic constraints are considered:

1) Task *C* cannot start if task *D* is done. In Figure 3, a possible FA-based model is shown for this constraint.

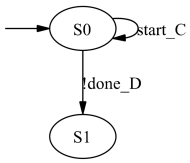


Fig. 3: FA representation of the first dynamic constraint.

2) If *A* is done before *B* and *C* has not yet started, then *C* should start immediately after *B* is done (Figure 4).

In this context, the term *immediately* denotes a strict precedence relation, ensuring no other tasks start or are done between the specified ones.

In addition to these five assembly tasks, the process includes a repeatable task, *F*, representing a *screwing operation*. This task is modeled as a loop automaton, as shown in Figure 1 (b), and its repeatability is governed by a dynamic constraint consisting of two parts in sequence. The first part is an *if-clause*: if task A is done immediately after B is done, then F must start immediately and be done before C and D start. The second part applies unconditionally: F must start immediately after C and D are done and must be done before E starts. Due to space limitations, the FA of this constraint is not shown.

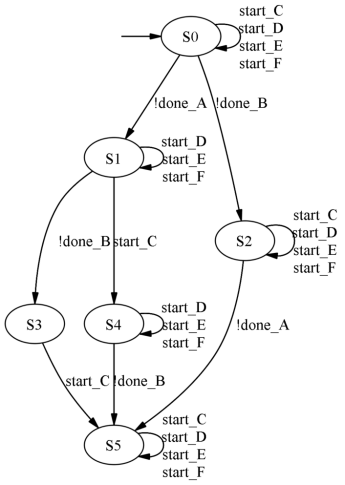


Fig. 4: FA representation of the second dynamic constraint.

Using SUPREMICA's *Analyzer* tool, the tasks and constraints were synchronized, and the resulting automaton is shown in Figure 5. The only marked state, q_{29} highlighted with a green circle at the bottom, represents that all tasks have been completed. The automaton consists of 33 states and 45 transitions and contains a blocking state, q_{14}, indicated by a dashed red circle. This state is blocking because there are no outgoing transitions from it that respect the defined constraints, making it impossible to reach the marked state.

By synthesizing a controllable and non-blocking supervisor using the monolithic algorithm and applying language equivalence minimization, the automaton shown in Figure 6 is obtained. The resulting supervisor consists of 25 states and 34 transitions. The synthesis process disables the blocking state q_{14}. Since the event leading to q_{14} is uncontrollable, the preceding state is also removed. If that state were also uncontrollable, the supervisor would continue removing previous states, tracing back to the first controllable state.

As a result, the synthesized supervisor ensures that the system follows only feasible assembly sequences, guaranteeing successful completion of the assembly process.

For large-scale assembly systems, SUPREMICA's monolithic algorithm may not be able to synthesize a supervisor due to state-space explosion. Scalability can be improved with alternative synthesis techniques such as BDD-based methods for compact state-space representation, and compositional abstraction-based synthesis, which partitions the system into subsystems with local supervisors combined into a global solution [12].

V. CONCLUSIONS

The proposed work enhances the modeling of dependencies and constraints between assembly tasks by incorporating not only static precedence constraints but also more detailed dependencies that reflect various process requirements. This feature meets the needs of modern assembly processes, which require greater flexibility to handle complex dependencies and dynamic production conditions.

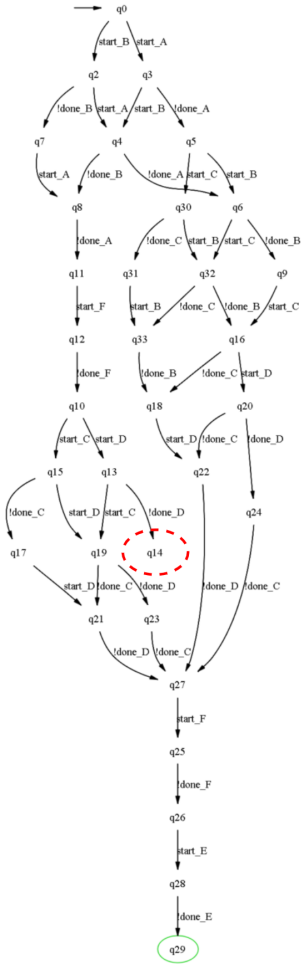


Fig. 5: Synchronized system automaton.

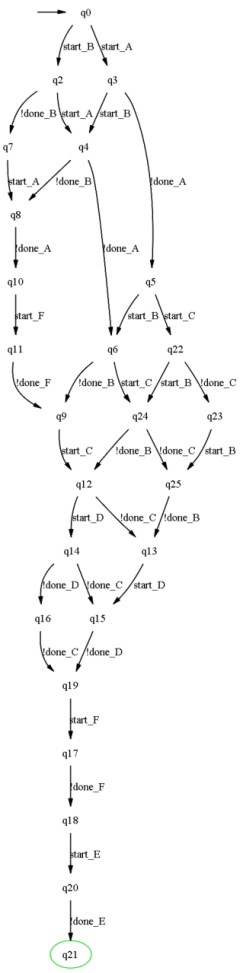


Fig. 6: Minimized supervisor automaton.

Using SCT and synthesizing a controllable, non-blocking, and minimally restrictive supervisor in SUPREMICA, *all* feasible assembly sequences for process completion were generated. These sequences can serve as a basis for further optimization to achieve specific objectives or for developing an exception management system to handle disruptions.

A challenge of this approach is the complexity of modeling certain dynamic constraints, which may require intricate formulations. A potential solution is to express such constraints as Boolean expressions and automatically convert them into finite automata, streamlining the modeling process.

REFERENCES

[1] D. Rossit, F. Tohmé, and M. Frutos, "An industry 4.0 approach to assembly line resequencing." *Int J Adv Manuf Technol*, vol. 105, p. 3619–3630, 2019.

[2] S. Hu, J. Ko, L. Weyand, H. ElMaraghy, T. Lien, Y. Koren, H. Bley, G. Chryssolouris, N. Nasr, and M. Shpitalni, "Assembly system design and operations for product variety," *CIRP Annals*, vol. 60, no. 2, pp. 715–733, 2011.

[3] F. S. Fogliatto, G. J. da Silveira, and D. Borenstein, "The mass customization decade: An updated review of the literature," *International Journal of Production Economics*, vol. 138, no. 1, pp. 14–25, 2012.

[4] A. C. Sanderson, L. S. Homem de Mello, and H. Zhang, "Assembly sequence planning," *AI Magazine*, vol. 11, no. 1, p. 62, Mar. 1990.

[5] Z. Han, Y. Wang, and D. Tian, "Ant colony optimization for assembly sequence planning based on parameters optimization," *Frontiers of Mechanical Engineering*, vol. 16, no. 2, pp. 393–409, 2021.

[6] H. Lv and C. Lu, "An assembly sequence planning approach with a discrete particle swarm optimization algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 50, no. 5, pp. 761–770, 2010.

[7] A. Adel, "Future of industry 5.0 in society: human-centric solutions, challenges and prospective research areas," *Journal of Cloud Computing*, vol. 11, no. 1, 2022.

[8] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer Publishing Company, Incorporated, 2010.

[9] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[10] K. Akesson, M. Fabian, H. Flordal, and R. Malik, "Supremica - an integrated environment for verification, synthesis and simulation of discrete event systems," in *2006 8th International Workshop on Discrete Event Systems*, 2006, pp. 384–385.

[11] B. Bonafilia, P. Carlsson, S. Nilsson, and M. Fabian, "Robust manual control of a manufacturing system using supervisory control theory," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 748–753, 2014, 19th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016417040

[12] R. Malik, S. Mohajerani, and M. Fabian, "A survey on compositional algorithms for verification and synthesis in supervisory control," *Discrete Event Dynamic Systems*, vol. 33, no. 3, pp. 279–340, Sep 2023. [Online]. Available: https://doi.org/10.1007/s10626-023-00378-8