ASML Embedded Systems Innovation BY TNO STW Enabling new technology RobustCPS

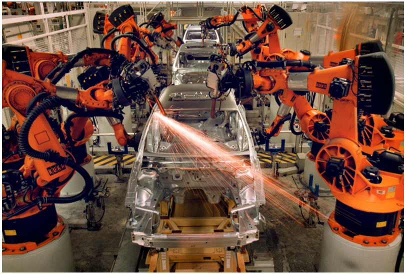
Compositional Specification of Functionality and Timing of Manufacturing Systems

Bram van der Sanden, João Bastos, Jeroen Voeten, Marc Geilen, Michel Reniers, Twan Basten, Johan Jacobs, and Ramon Schiffelers

TU/e Technische Universiteit Eindhoven University of Technology

September 16, 2016

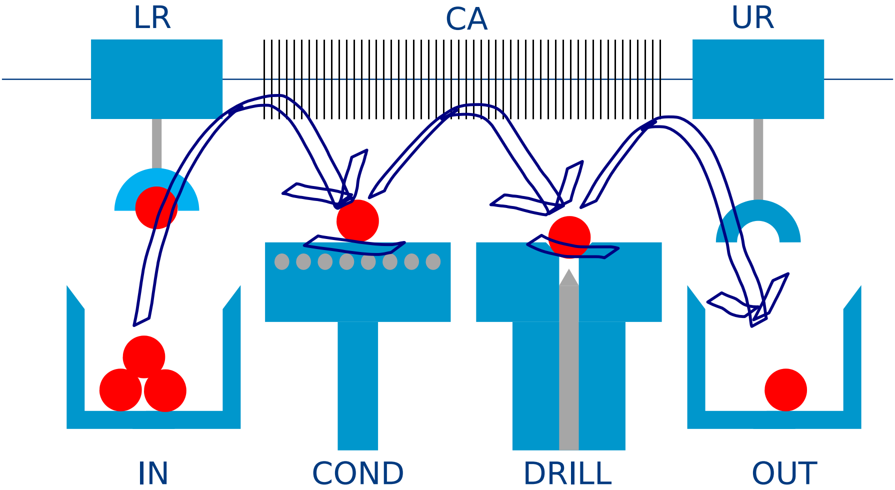
Manufacturing Systems



Goal:

- Formal modeling approach for manufacturing systems with Compositional Specification of Functionality and Timing.

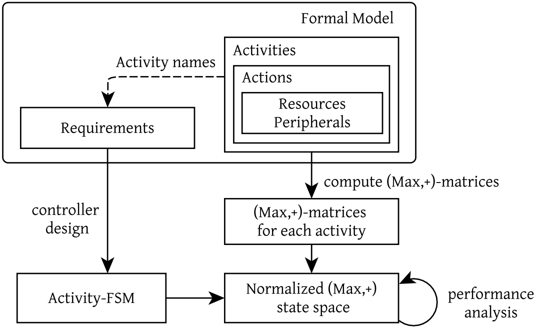
Example Manufacturing System



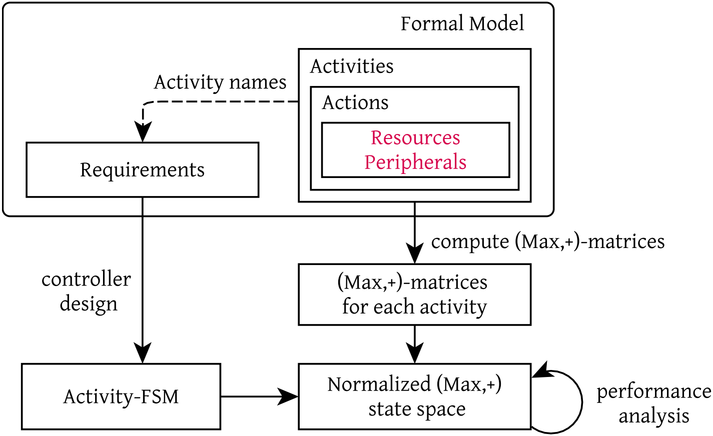
Technische Universiteit **Eindhoven** University of Technology

Formal Modeling Approach

Modeling and Analysis to find a throughput-optimal safe controller for nominal behavior.



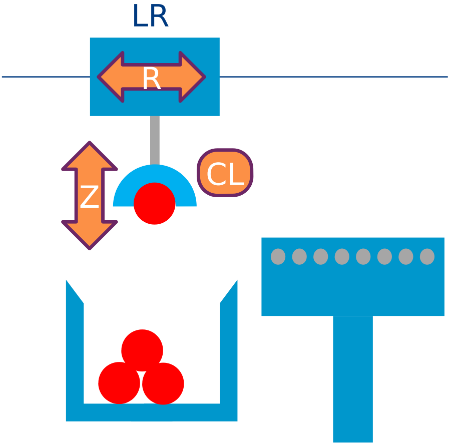
Formal Model: Resources and Peripherals



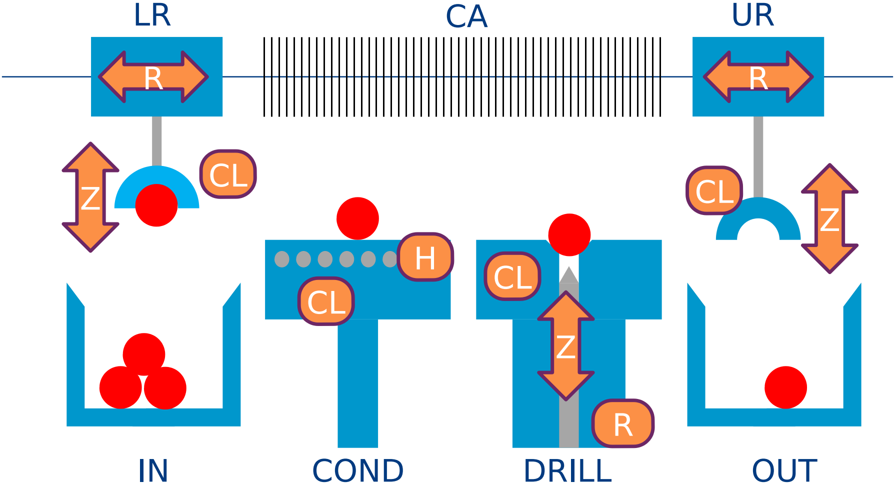
Formal Model: Resources and Peripherals

- Set of peripherals that can execute a number of actions.
- Peripherals are aggregated into resources.

- Resource Load Robot LR has peripherals motor R, motor Z, clamp CL.



Formal Model: Resources and Peripherals



Formal Model: System behavior

Resources and peripherals: model the system **components.**

System **behavior** is modeled on three levels:

1. Actions executed by peripherals.

Formal Model: System behavior

Resources and peripherals: model the system **components.**

System **behavior** is modeled on three levels:

1. Actions executed by peripherals.
2. Activities to describe scenarios of end-to-end **deterministic behavior.**

   An activity consists of a set of actions and dependencies among these actions.

Formal Model: System behavior

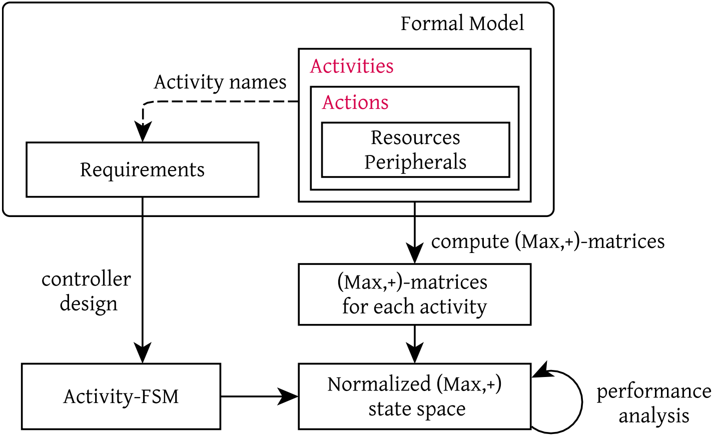
Resources and peripherals: model the system **components.**

System **behavior** is modeled on three levels:

1. Actions executed by peripherals.
2. Activities to describe scenarios of end-to-end **deterministic behavior.**

   An activity consists of a set of actions and dependencies among these actions.
3. Activity sequences describe **orderings of activities.**

Formal Model: Activities and Actions



Formal Model: Activities and Actions

Activities in the Twilight system:

Operations on a product:

- Condition
- Drill

Moving a product:

- LR_PickFromInput
- LR_PickFromCond
- LR_PickFromDrill
- LR_PutOnCond
- LR_PutOnDrill
- UR_PickFromCond
- UR_PickFromDrill
- UR_PutOnCond
- UR_PutOnDrill
- UR_PutOnOutput

Formal Model: Activities and Actions

Activities in the Twilight system:

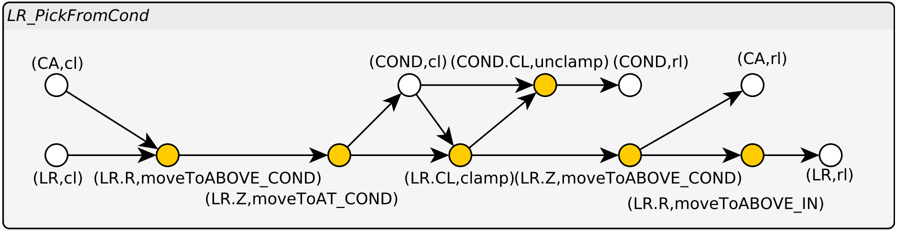
Operations on a product:

- Condition
- Drill

Moving a product:

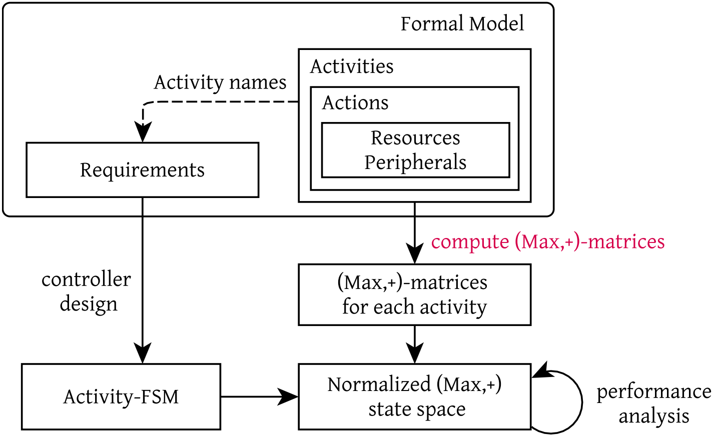
- LR_PickFromInput
- LR_PickFromCond
- LR_PickFromDrill
- LR_PutOnCond
- LR_PutOnDrill
- UR_PickFromCond
- UR_PickFromDrill
- UR_PutOnCond
- UR_PutOnDrill
- UR_PutOnOutput

Formal Model: Activities and Actions

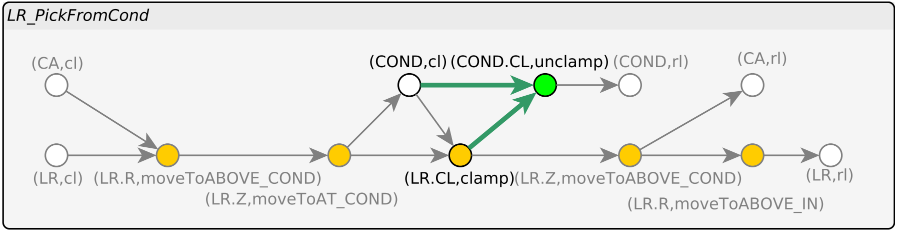


- Function of activity:
  Move LR to COND, pick up the ball, move back to home position
- Actions: cl claim, rl release, unclamp, clamp, moves
- Involved resources:
  Collision Area (CA), Load Robot (LR), Conditioner (COND)

Formal Model: Timing



Formal Model: Timing

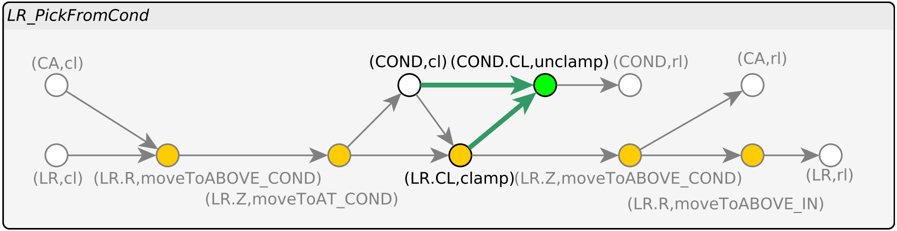


Two essential characteristics in activity execution:

1. synchronization: waits for incoming dependencies to finish

Technische Universiteit Eindhoven University of Technology

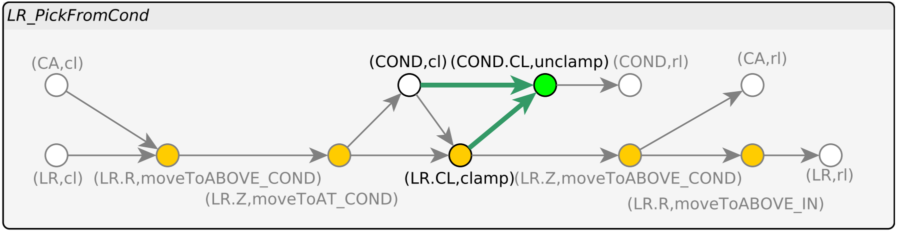
Formal Model: Timing



Two essential characteristics in activity execution:

1. synchronization: waits for incoming dependencies to finish
2. delay: duration of execution before completion

Formal Model: Timing

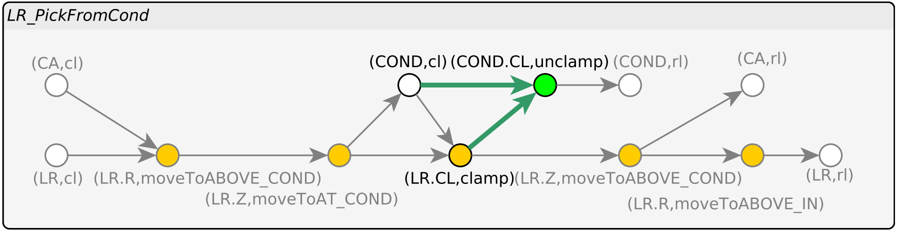


Two essential characteristics in activity execution:

1. synchronization: waits for incoming dependencies to finish
2. delay: duration of execution before completion

These characteristics correspond well to the (max,+) operators max and addition in (max,+) algebra.

Formal Model: Timing



Two essential characteristics in activity execution:

1. synchronization: waits for incoming dependencies to finish
2. delay: duration of execution before completion

These characteristics correspond well to the (max,+) operators max and addition in (max,+) algebra.

We use this algebra to capture the timing behavior of the activity in a (max,+) matrix.

Formal Model: Timing

- (max,+) matrix: $M_{LR\_PickFromCond} = \begin{bmatrix} .. & .. & .. \\ .. & .. & .. \\ .. & .. & .. \end{bmatrix}$

Formal Model: Timing

- (max,+) matrix: $M_{LR\_PickFromCond} = \begin{bmatrix} .. & .. & .. \\ .. & .. & .. \\ .. & .. & .. \end{bmatrix}$
- resource availability vector: $\gamma_{begin} = \begin{bmatrix} \gamma_{begin}(\mathsf{CA}) \\ \gamma_{begin}(\mathsf{LR}) \\ \gamma_{begin}(\mathsf{COND}) \end{bmatrix}$

Technische Universiteit **Eindhoven** University of Technology

Formal Model: Timing

- (max,+) matrix: $M_{LR\_PickFromCond} = \begin{bmatrix} .. & .. & .. \\ .. & .. & .. \\ .. & .. & .. \end{bmatrix}$
- resource availability vector: $\gamma_{begin} = \begin{bmatrix} \gamma_{begin}(\mathsf{CA}) \\ \gamma_{begin}(\mathsf{LR}) \\ \gamma_{begin}(\mathsf{COND}) \end{bmatrix}$
- New vector γ_{end} by matrix multiplication in (max,+) algebra:

$$\underbrace{\begin{bmatrix} \gamma_{end}(\mathsf{CA}) \\ \gamma_{end}(\mathsf{LR}) \\ \gamma_{end}(\mathsf{COND}) \end{bmatrix}}_{\gamma_{end}} = \underbrace{\begin{bmatrix} .. & .. & .. \\ .. & .. & .. \\ .. & .. & .. \end{bmatrix}}_{M_{LR\_PickFromCond}} \otimes \underbrace{\begin{bmatrix} \gamma_{begin}(\mathsf{CA}) \\ \gamma_{begin}(\mathsf{LR}) \\ \gamma_{begin}(\mathsf{COND}) \end{bmatrix}}_{\gamma_{begin}}$$

Formal Model: Timing

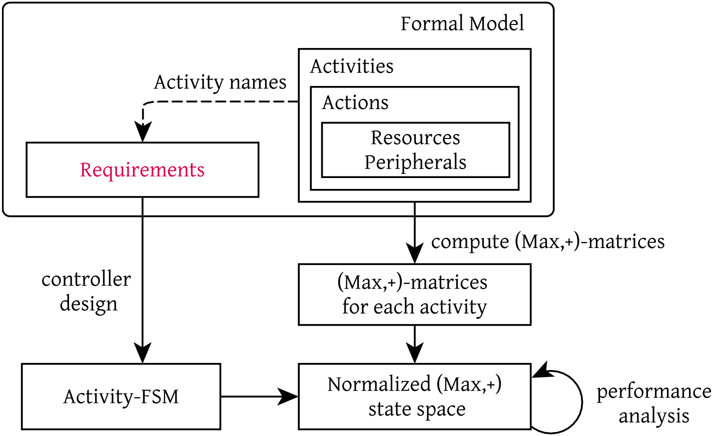
Now consider an activity sequence, for instance:

$$\text{Condition} \; ; \; \text{LR\_PickFromCond}.$$

Then, given begin system state γ_{begin},
the end system state γ_{end} is given by:

$$\gamma_{end} = M_{\text{LR\_PickFromCond}} \otimes M_{\text{Condition}} \otimes \gamma_{begin}.$$

Formal Model: Requirements



Formal Model: Requirements

What we have so far:

- Activities: specify scenarios of **deterministic behavior.**
- Activity sequences: to describe **orderings of activities.**

Formal Model: Requirements

What we have so far:

- Activities: specify scenarios of **deterministic behavior.**
- Activity sequences: to describe **orderings of activities.**

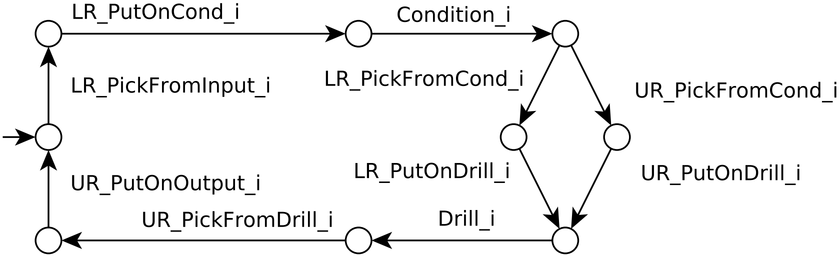
Not all orderings are however valid. We need requirements on allowed orderings to enforce:

- No product collisions at product locations.
- No collisions of robots.
- Life cycle of products.

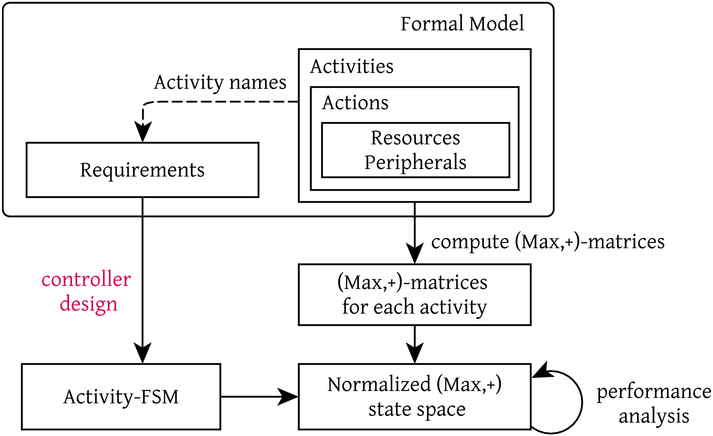
Formal Model: Requirements

Life cycle automaton: each product follows the same life cycle.

- We restrict to forward moves only.
- Scheduling freedom between LR and UR.



Controller Design



Controller Design

Formal Model specifies:

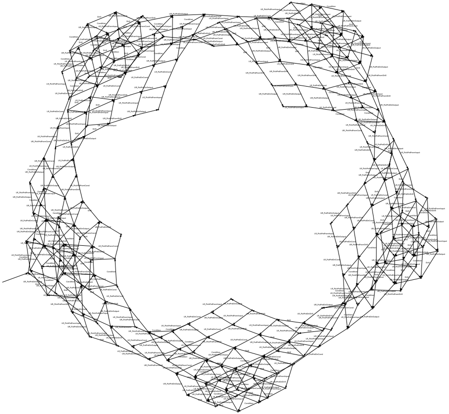
- Activities in system.
- Requirements on allowed activity orderings.

Controller Design:

- Design a controller that respects the requirements.
- For our example, we use supervisory controller synthesis to obtain an Activity-FSM of all allowed activity sequences.
- Activity-FSM is guaranteed to be deadlock-free and functionally correct with respect to the requirements.
- In this step we can **abstract from the timing !**

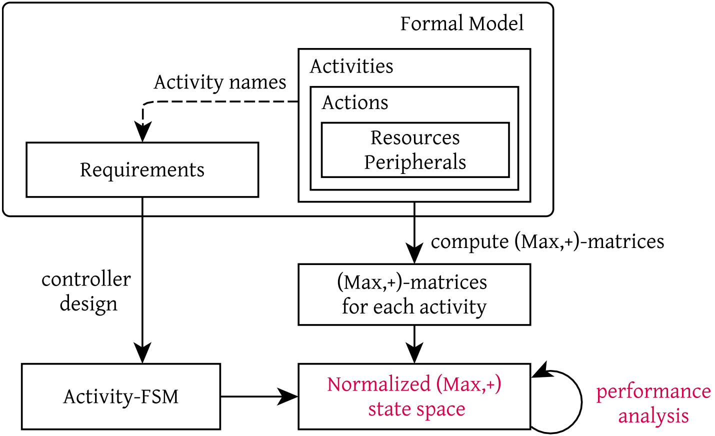
Controller Design

Activity-FSM after synthesis (245 locations, 510 transitions):



Models all allowed activity sequences.

Performance Analysis



Performance Analysis

What we have achieved so far:

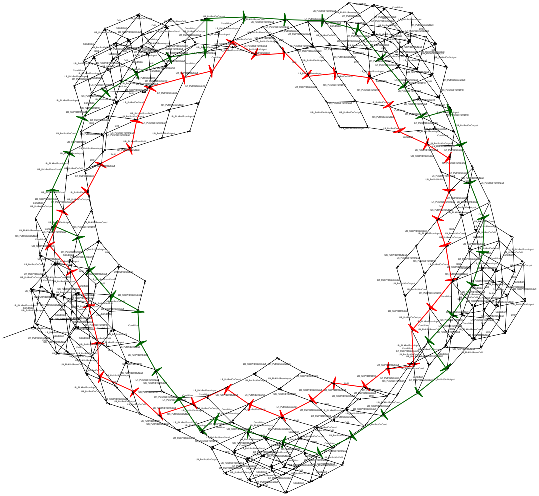
- Activity-FSM that models all allowed activity sequences.
- Timing matrix for each activity.

Now:

- Combine the (max,+) timing matrices and Activity-FSM.
- Explore normalized (max,+) state space, which is finite.
- Find optimal controller by performance analysis on traces in the state space.

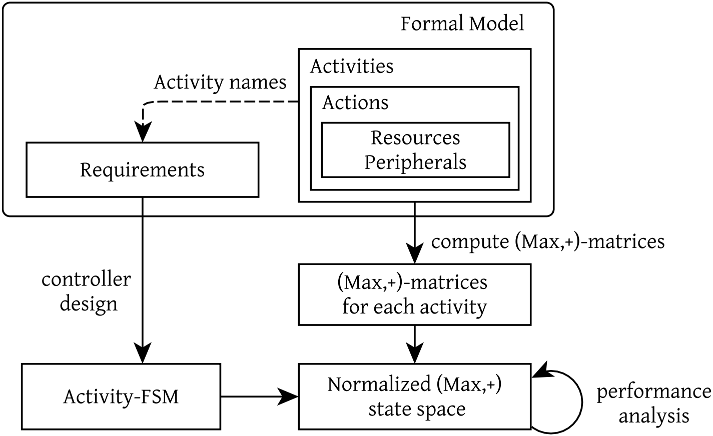
Performance Analysis on Twilight System

Using performance analysis algorithm to find guaranteed throughput and optimal controller.



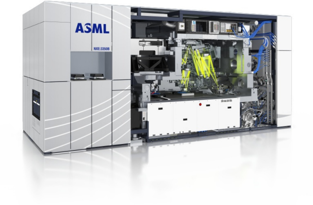
Recap: Formal Modeling Approach

Compositional Specification of Functionality and Timing.



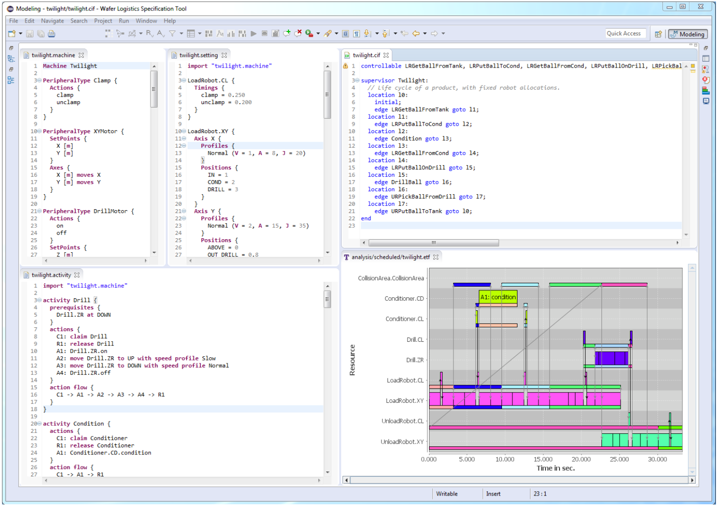
Industrial use case: ASML

- Semantic underpinning of wafer handling specification tool.
- Model resources, peripherals, actions, activities.
- But also: motion paths of robots, symbolic positions.
- Calculate timing of move actions from the specification.
- Current research: specification of wafer logistics for nominal behavior.



ASML

Industrial use case: ASML



twilight.machine

```
Machine Twilight

PeripheralType Clamp {
  Actions {
    clamp
    unclamp
  }
}

PeripheralType XYMotor {
  SetPoints {
    X [m]
    Y [m]
  }
  Axes {
    X [m] moves X
    Y [m] moves Y
  }
}

PeripheralType DrillMotor {
  Actions {
    on
    off
  }
  SetPoints {
    Z [m]
```

twilight.setting

```
import "twilight.machine"

LoadRobot.CL {
  Timings {
    clamp = 0.250
    unclamp = 0.200
  }
}

LoadRobot.XY {
  Axis X {
    Profiles {
      Normal (V = 1, A = 8, J = 20)
    }
    Positions {
      IN = 1
      COND = 2
      DRILL = 3
    }
  }
  Axis Y {
    Profiles {
      Normal (V = 2, A = 15, J = 35)
    }
    Positions {
      ABOVE = 0
      OUT DRILL = 0.8
```

twilight.cif

```
controllable LRGetBallFromTank, LRPutBallToCond, LRGetBallFromCond, LRPutBallOnDrill, LRPickBal

supervisor Twilight:
  // Life cycle of a product, with fixed robot allocations.
  location l0:
    initial;
    edge LRGetBallFromTank goto l1;
  location l1:
    edge LRPutBallToCond goto l2;
  location l2:
    edge Condition goto l3;
  location l3:
    edge LRGetBallFromCond goto l4;
  location l4:
    edge LRPutBallOnDrill goto l5;
  location l5:
    edge DrillBall goto l6;
  location l6:
    edge URPickBallFromDrill goto l7;
  location l7:
    edge URPutBallToTank goto l0;
end
```

twilight.activity

```
import "twilight.machine"

activity Drill {
  prerequisites {
    Drill.ZR at DOWN
  }
  actions {
    C1: claim Drill
    R1: release Drill
    A1: Drill.ZR.on
    A2: move Drill.ZR to UP with speed profile Slow
    A3: move Drill.ZR to DOWN with speed profile Normal
    A4: Drill.ZR.off
  }
  action flow {
    C1 -> A1 -> A2 -> A3 -> A4 -> R1
  }
}

activity Condition {
  actions {
    C1: claim Conditioner
    R1: release Conditioner
    A1: Conditioner.CD.condition
  }
  action flow {
    C1 -> A1 -> R1
```

Future work

Research directions:

- Size of state space: modular synthesis techniques.
- More intuitive requirement modeling using state-based expressions.
- Uncontrollable behavior: extend formalism, new throughput analysis and optimization techniques.

Conclusions: Formal Modeling Approach

- Compositional Specification of Functionality and Timing.
- Functionality:
  - Describe deterministic behavior using **activities.**
  - Controller choices determine the **order of these activities.**
  - Separation between deterministic behavior and nondeterministic behavior.
  - Controller design on activity level, **abstraction from timing.**
- Timing:
  - Fixed timing for actions.
  - Timing behavior of activities captured in **matrices.**
  - System behavior: **max-plus state space**, timing analysis.

Technische Universiteit
Eindhoven
University of Technology