CHANGE AND DELAY CONTRACTS FOR HYBRID SYSTEM COMPONENT VERIFICATION

Andreas Müller – andreas.mueller@jku.at
Werner Retschitzegger – werner.retschitzegger@jku.at
Wieland Schwinger – wieland.schwinger@jku.at

Johannes Kepler University, Linz
Department of Cooperative Information Systems
http://cis.jku.at/

Stefan Mitsch – smitsch@cs.cmu.edu
André Platzer - aplatzer@cs.cmu.edu

Carnegie Mellon University, Pittsburgh
Computer Science Department
http://www.ls.cs.cmu.edu

OVERVIEW

- Background
  - Cyber-Physical Systems
  - Component-based Verification

- Component-based Verification
  - Components
  - Contracts
  - Composition

- Evaluation
  - Implementation
  - Experiments

OVERVIEW

- Background
  - Cyber-Physical Systems
  - Component-based Verification

- Component-based Verification
  - Components
  - Contracts
  - Composition

- Evaluation
  - Implementation
  - Experiments

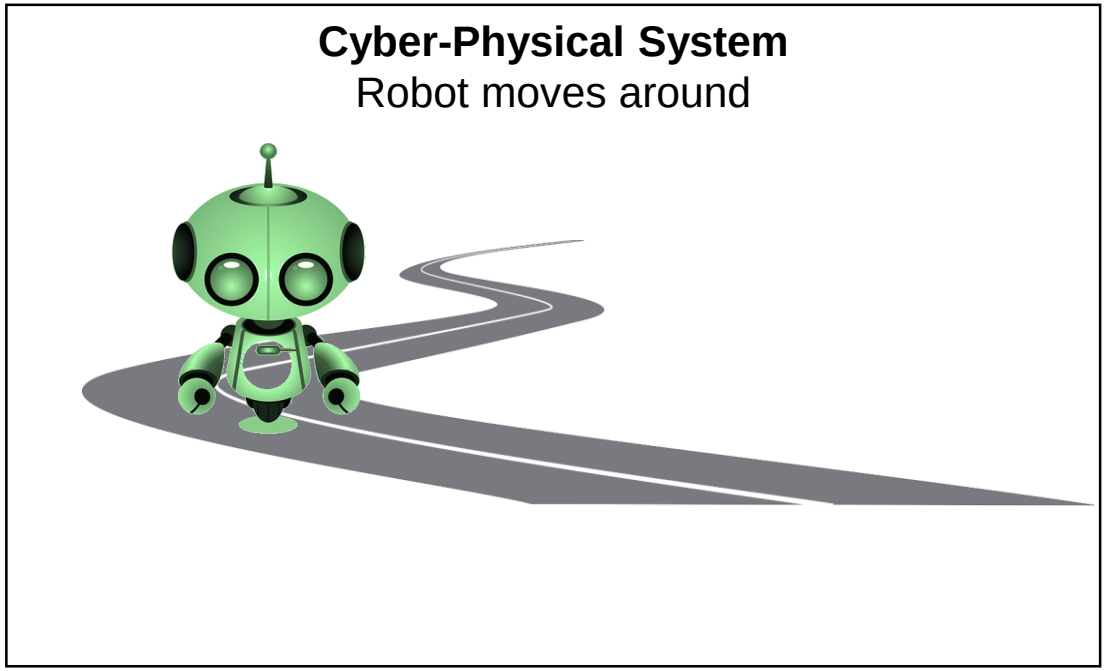
BACKGROUND – CYBER-PHYSICAL SYSTEMS

Cyber-physical systems (CPS)

- **Cyber** and **physical** capabilities

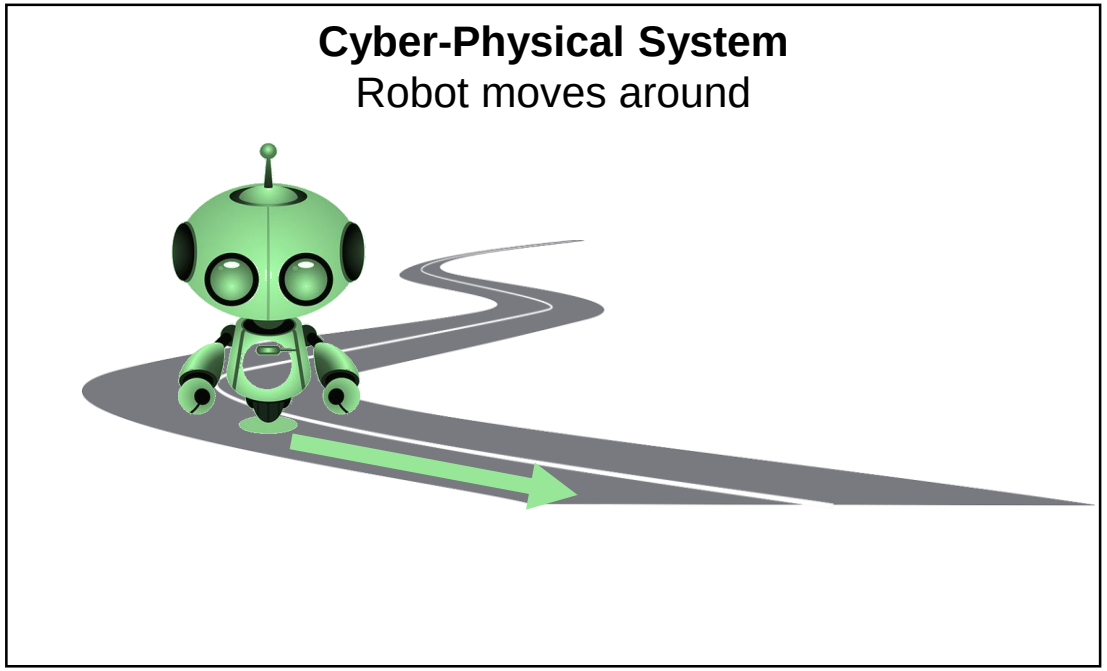
BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities



BACKGROUND – CYBER-PHYSICAL SYSTEMS

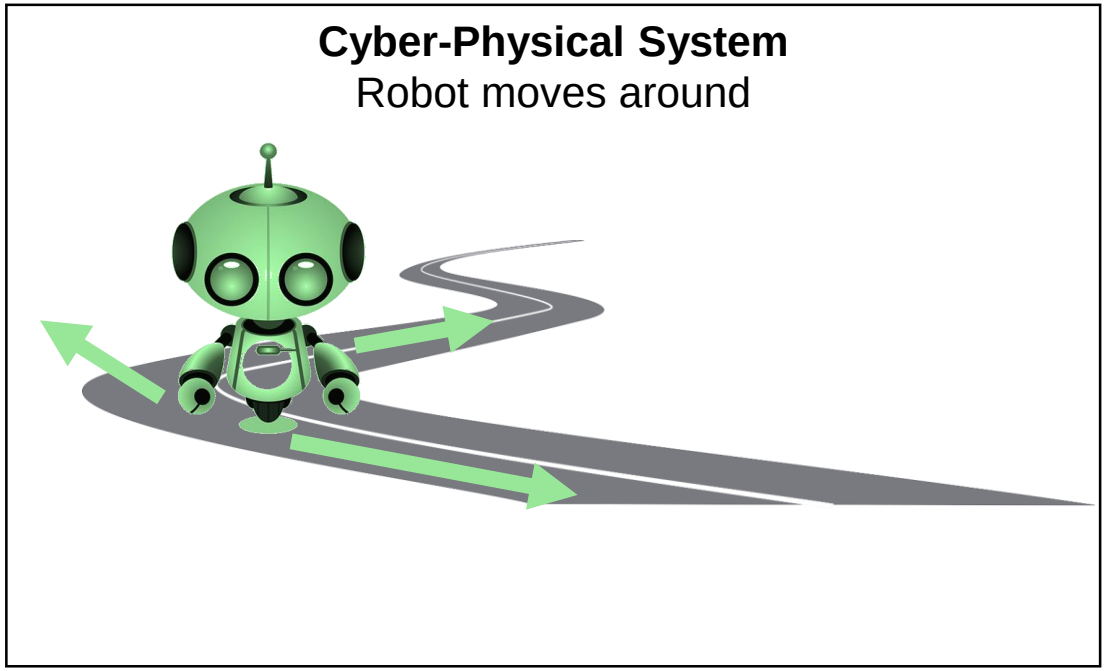
- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement



BACKGROUND – CYBER-PHYSICAL SYSTEMS

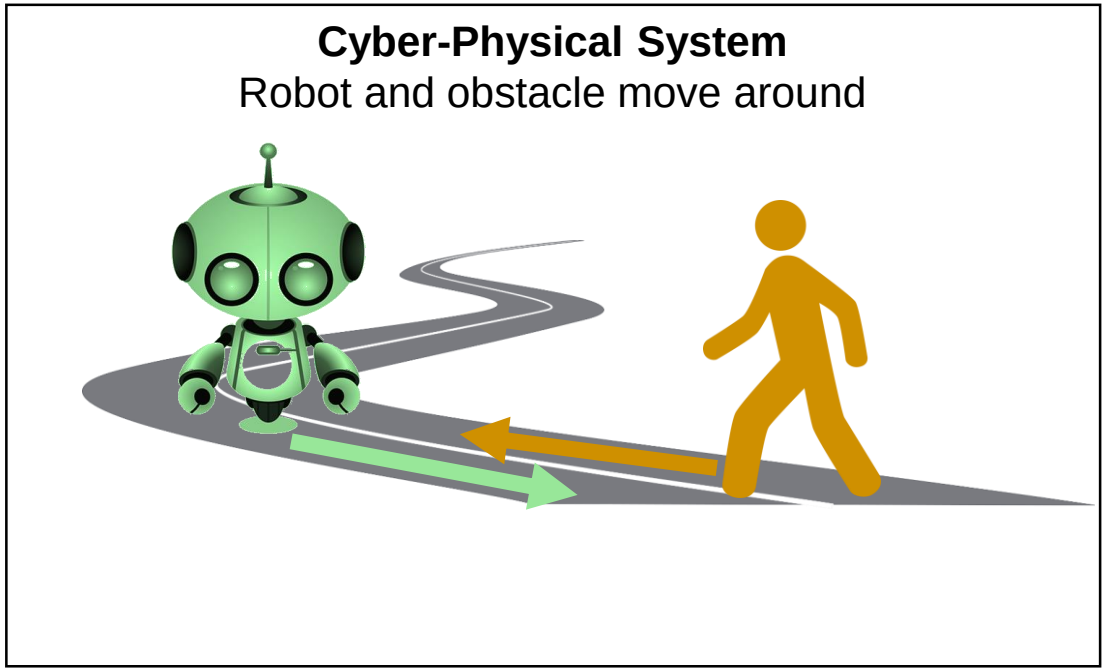
Cyber-physical systems (CPS)

- ☐ **Cyber** and **physical** capabilities
- ☐ Continuous physical-part, e.g., movement
- ☐ Discrete cyber-part, e.g., steering



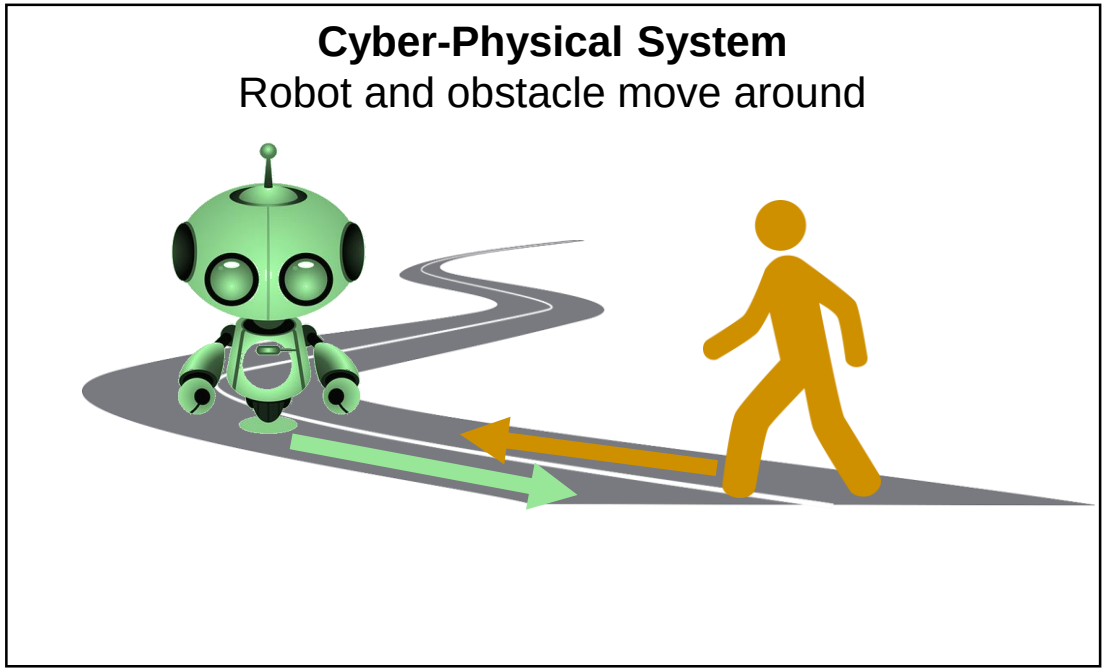
BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement
  - Discrete cyber-part, e.g., steering
- Complex cyber-physical systems are typically composed of multiple interacting subsystems



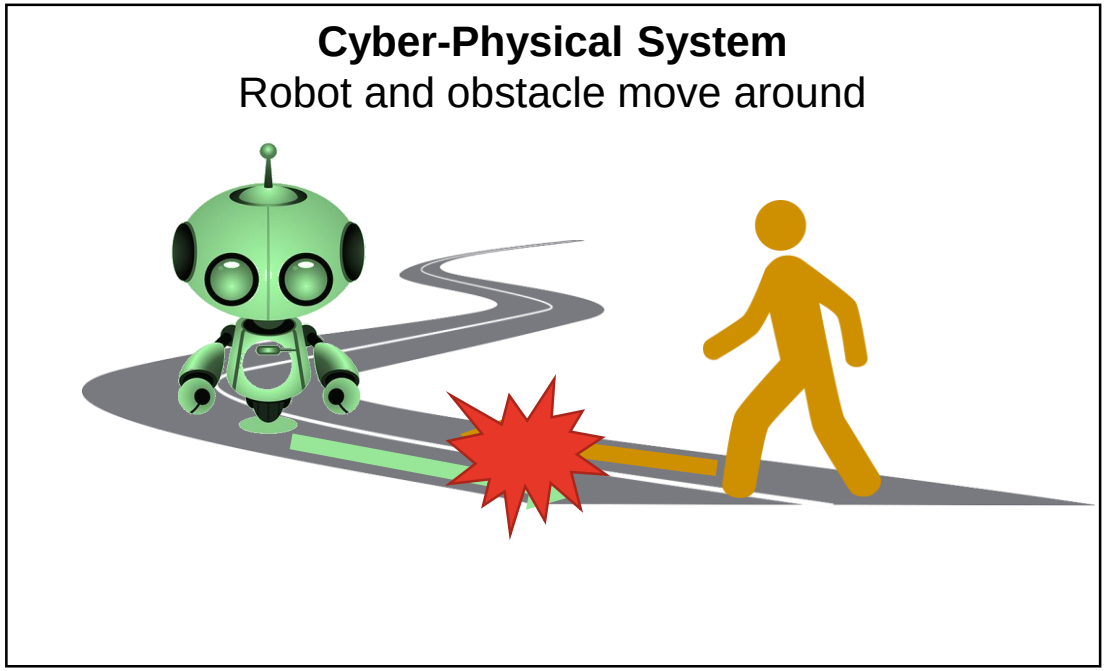
BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement
  - Discrete cyber-part, e.g., steering
- Complex cyber-physical systems are typically composed of multiple interacting subsystems
  - Often **safety-critical**!



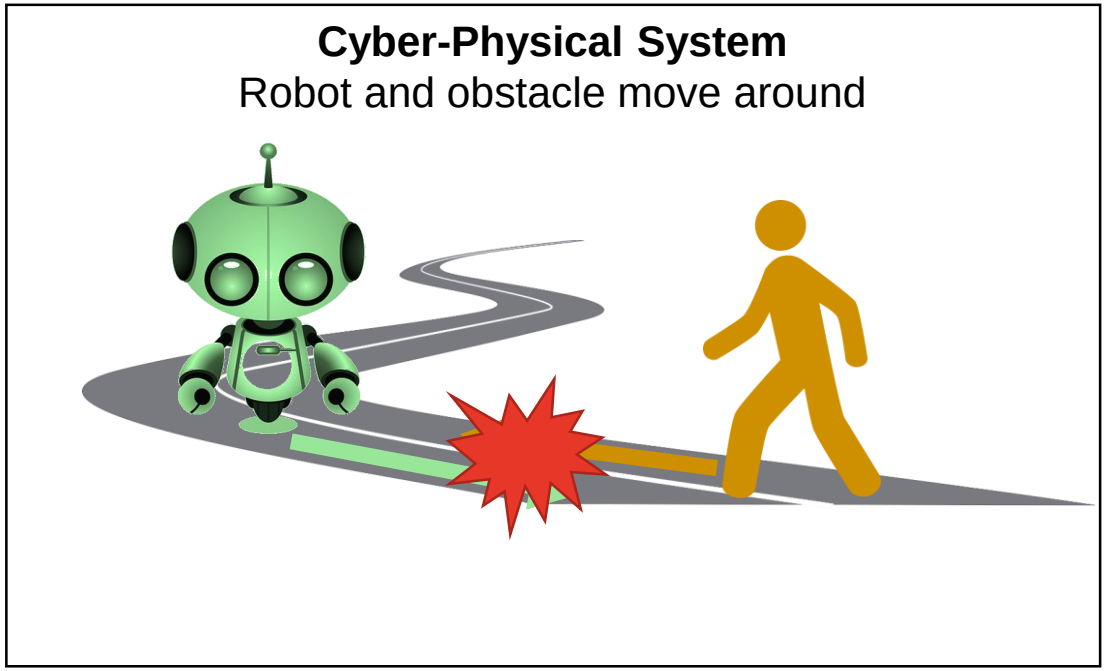
BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement
  - Discrete cyber-part, e.g., steering
- Complex cyber-physical systems are typically composed of multiple interacting subsystems
  - Often **safety-critical**!



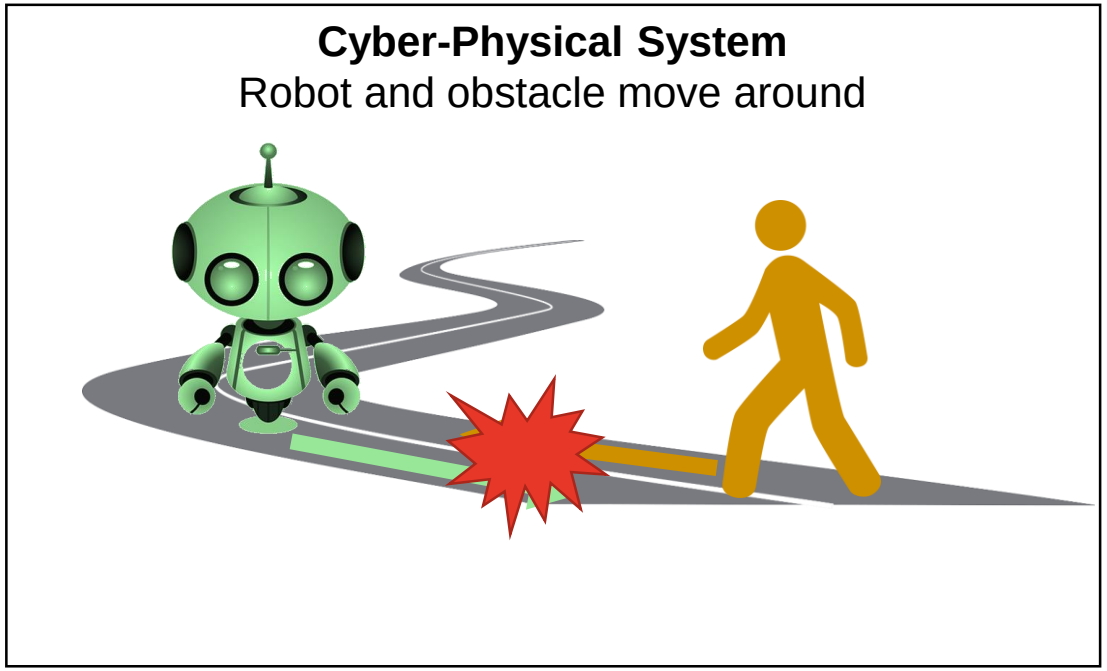
BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement
  - Discrete cyber-part, e.g., steering
- Complex cyber-physical systems are typically composed of multiple interacting subsystems
  - Often **safety-critical**!
- Model and analyze CPS: **Hybrid system models**



BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement
  - Discrete cyber-part, e.g., steering
- Complex cyber-physical systems are typically composed of multiple interacting subsystems
  - Often **safety-critical**!
- Model and analyze CPS: **Hybrid system models**
  - Hybrid programs: program notation for CPS
  - Safety Property: $\Phi \rightarrow [\alpha]\Psi$
    - System **contract**: Starting in Φ, each run of hybrid program α leads to a safe state Ψ
    - Verified using hybrid systems theorem prover – KeYmaera X
  - Analysis is challenging for **large monolithic models**

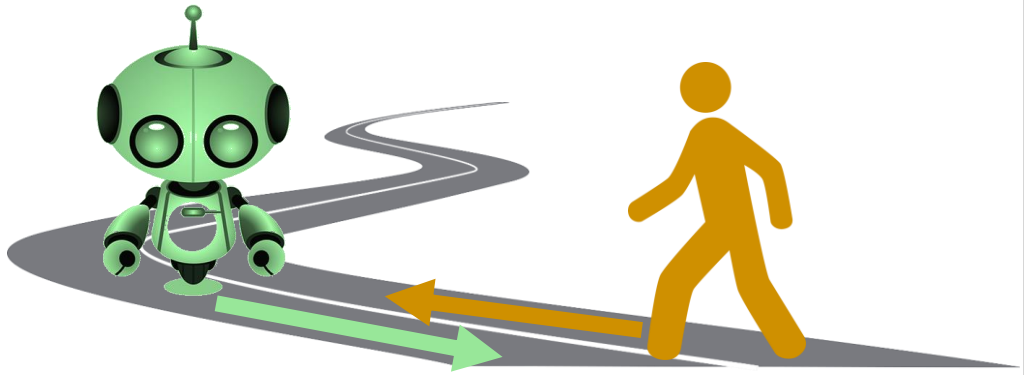


Idea: Component-based approach to hybrid system safety verification

BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement
  - Discrete cyber-part, e.g., steering
- Complex cyber-physical systems are typically composed of multiple interacting subsystems
  - Often **safety-critical**!
- Model and analyze CPS: **Hybrid system models**
  - Hybrid programs: program notation for CPS
  - Safety Property: $\Phi \rightarrow [\alpha]\Psi$
    - System **contract**: Starting in Φ, each run of hybrid program α leads to a safe state Ψ
    - Verified using hybrid systems theorem prover – KeYmaera X
  - Analysis is challenging for **large monolithic models**

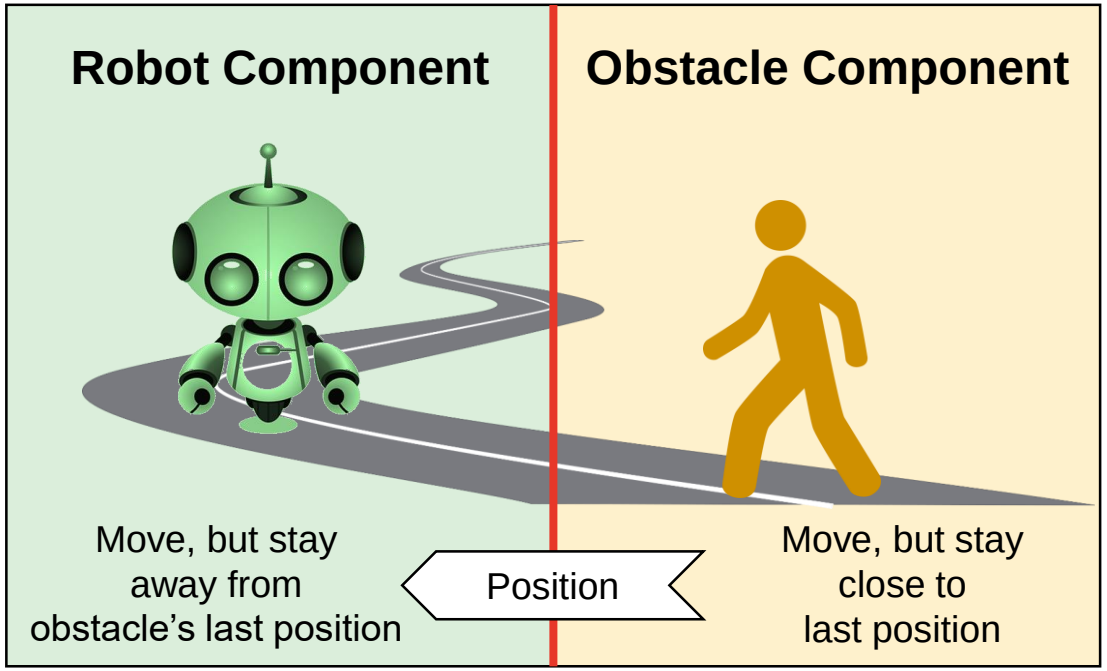
Cyber-Physical System
Robot moves, avoiding collisions with moving obstacle



Idea: Component-based approach to hybrid system safety verification

BACKGROUND – CYBER-PHYSICAL SYSTEMS

- Cyber-physical systems (CPS)
  - **Cyber** and **physical** capabilities
  - Continuous physical-part, e.g., movement
  - Discrete cyber-part, e.g., steering
- Complex cyber-physical systems are typically composed of multiple interacting subsystems
  - Often **safety-critical**!
- Model and analyze CPS: **Hybrid system models**
  - Hybrid programs: program notation for CPS
  - Safety Property: $\Phi \rightarrow [\alpha]\Psi$
    - System **contract**: Starting in Φ, each run of hybrid program α leads to a safe state Ψ
    - Verified using hybrid systems theorem prover – KeYmaera X
  - Analysis is challenging for **large monolithic models**



Idea: Component-based approach to hybrid system safety verification

BACKGROUND – COMPONENT-BASED VERIFICATION

Monolithic

Internal Behavior

System
(Component A ‖ Component B)

BACKGROUND – COMPONENT-BASED VERIFICATION

Monolithic

External Behavior

System Contract
(Contract A ∧ Contract B)

Internal Behavior

System
(Component A ‖ Component B)

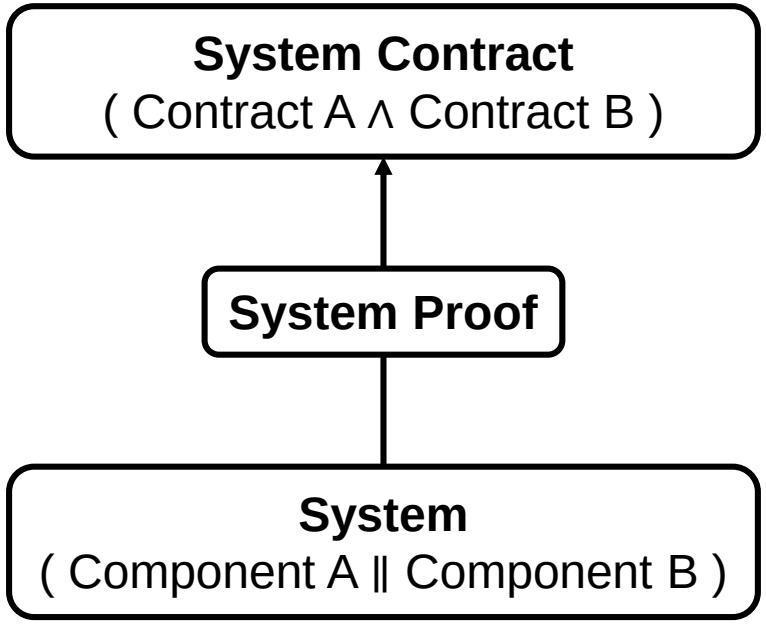
BACKGROUND – COMPONENT-BASED VERIFICATION

Monolithic

External Behavior

Formal Verification

Internal Behavior



BACKGROUND – COMPONENT-BASED VERIFICATION

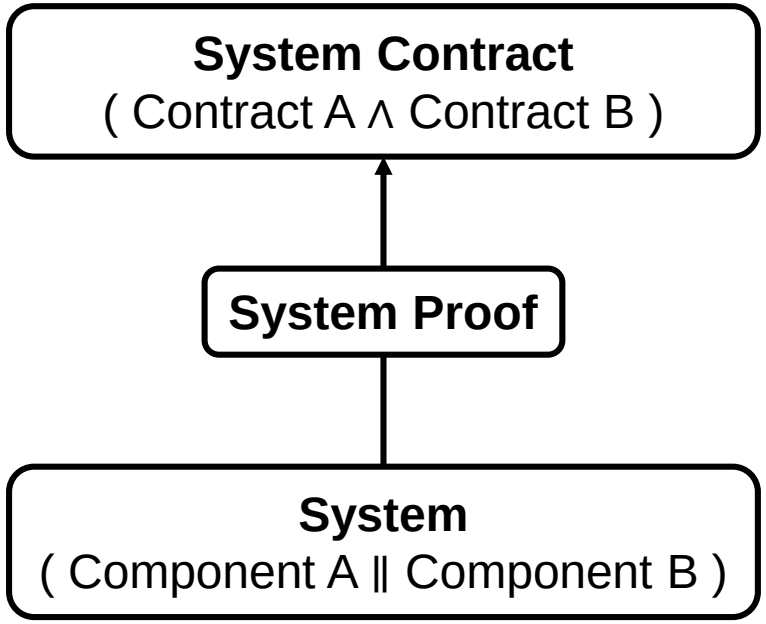
Monolithic

Component-based

External Behavior

Formal Verification

Internal Behavior



BACKGROUND – COMPONENT-BASED VERIFICATION

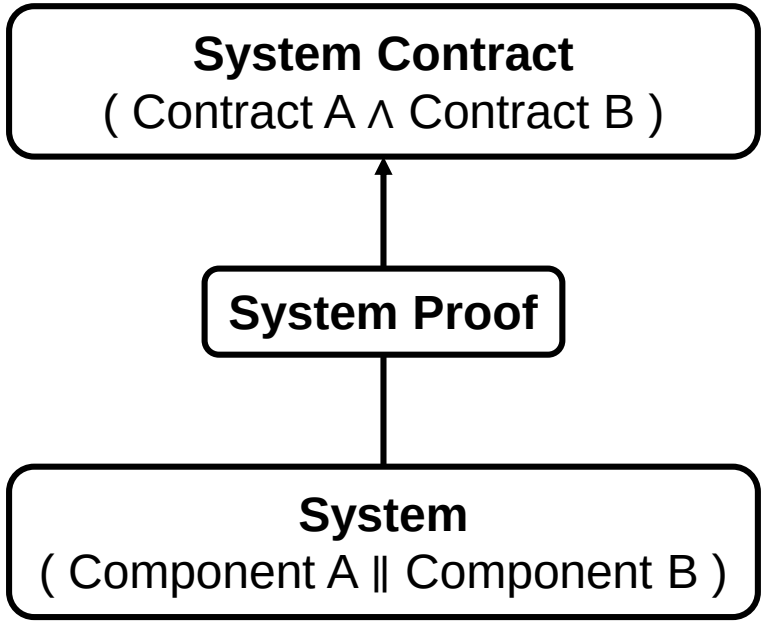
Monolithic

Component-based

External Behavior

Formal Verification

Internal Behavior



Component A

Component B

BACKGROUND – COMPONENT-BASED VERIFICATION

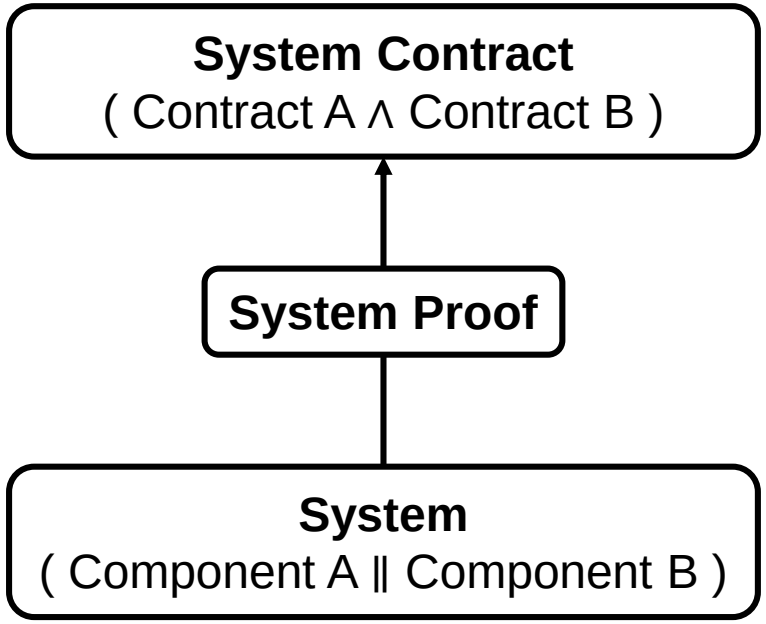
Monolithic

Component-based

External Behavior

Formal Verification

Internal Behavior



Contract A

Contract B

Component A

Component B

BACKGROUND – COMPONENT-BASED VERIFICATION

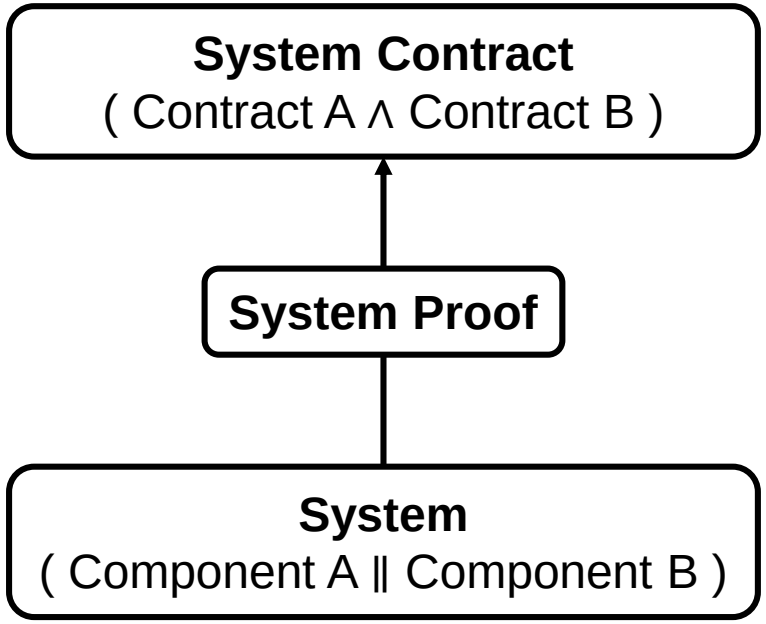
Monolithic

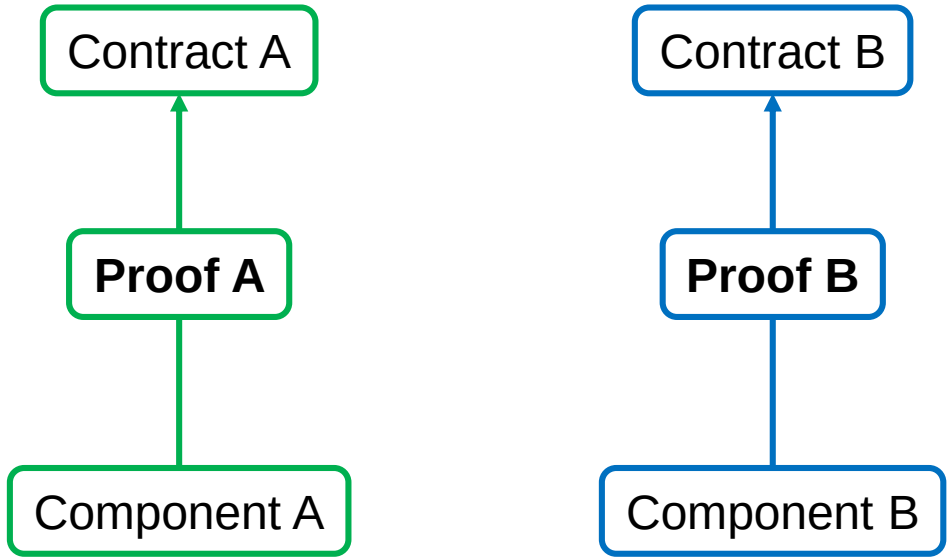
Component-based

External Behavior

Formal Verification

Internal Behavior





BACKGROUND – COMPONENT-BASED VERIFICATION

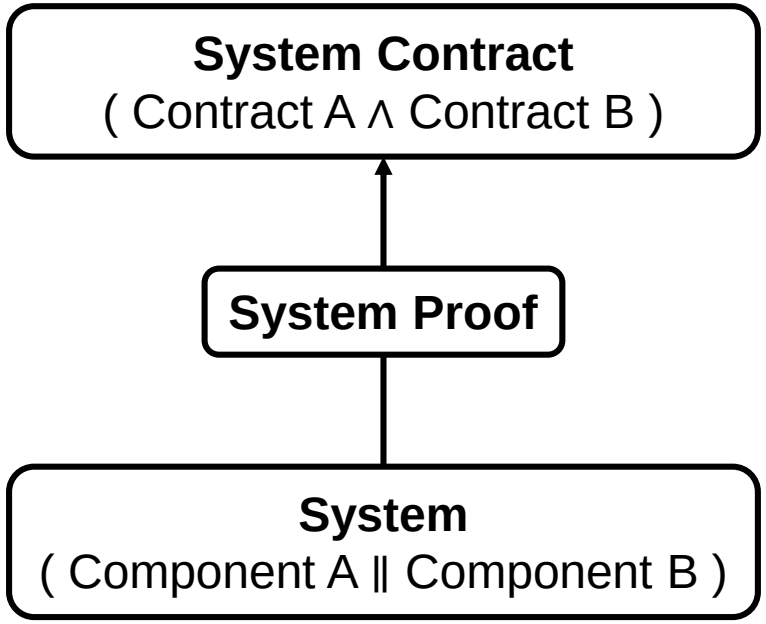
Monolithic

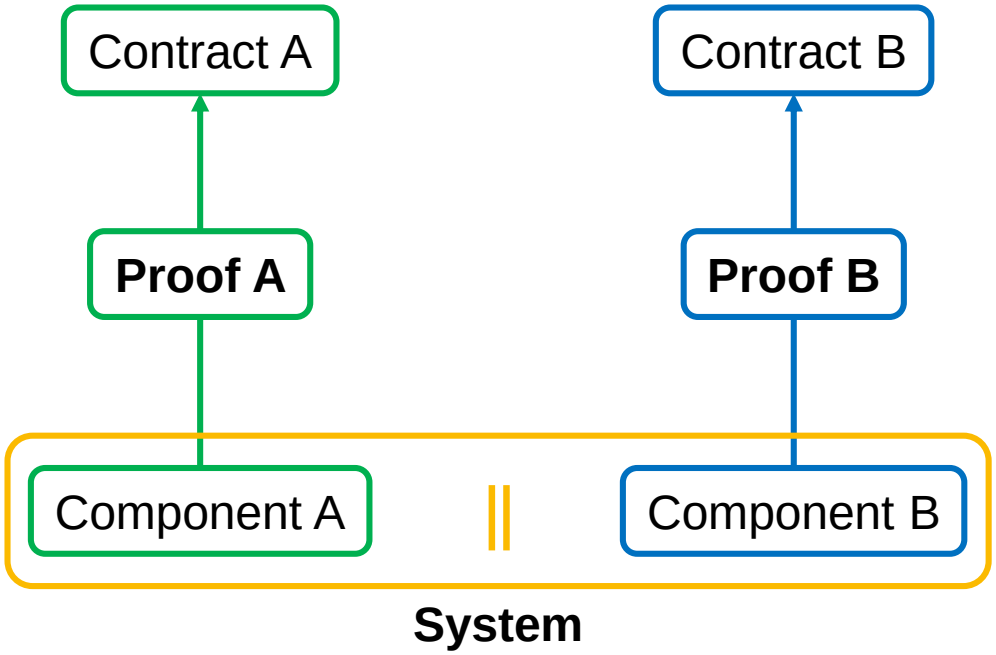
Component-based

External Behavior

Formal Verification

Internal Behavior





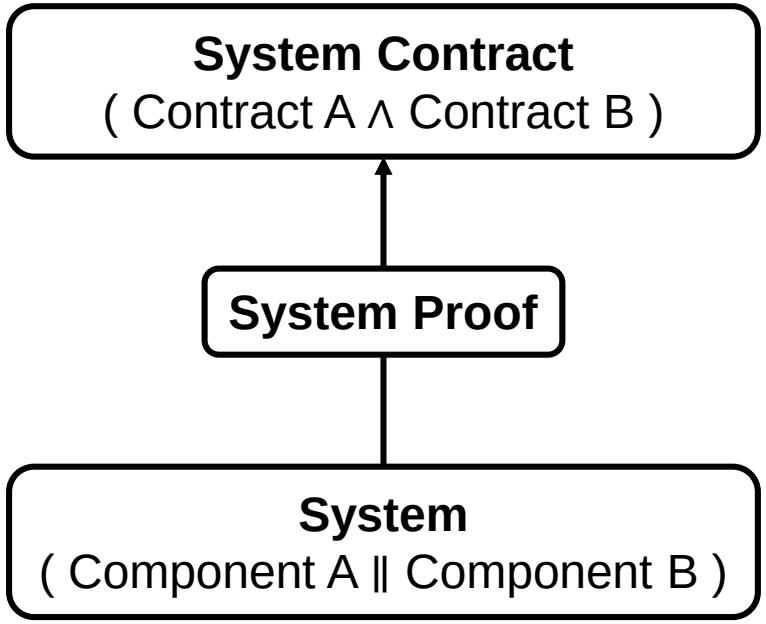
BACKGROUND – COMPONENT-BASED VERIFICATION

External Behavior

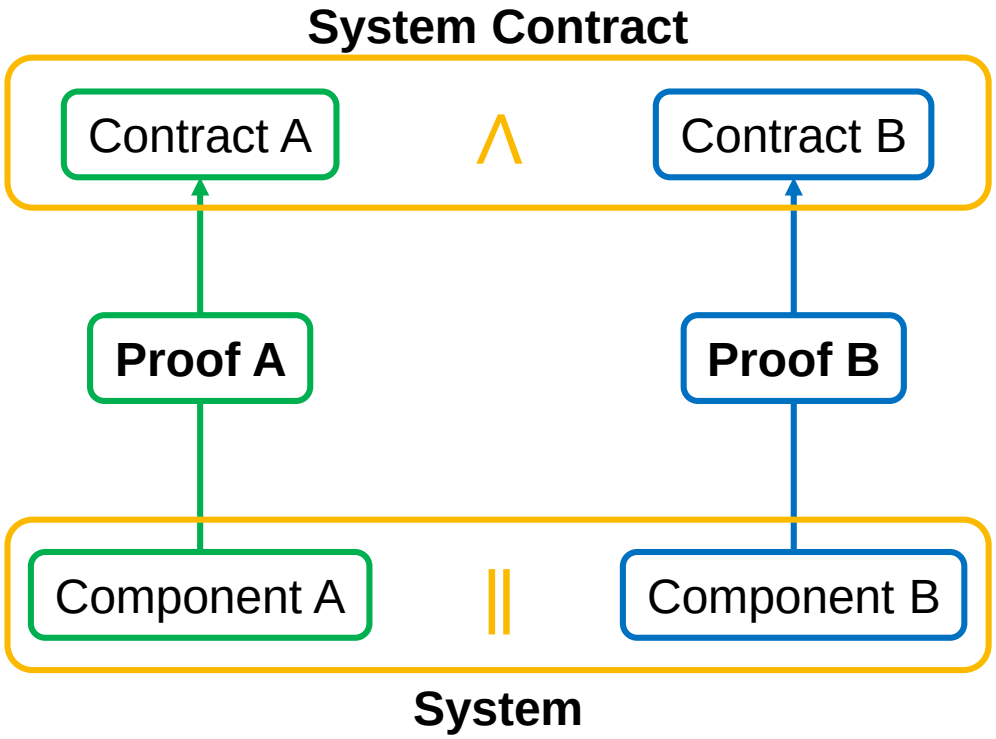
Formal Verification

Internal Behavior

Monolithic



Component-based



BACKGROUND – COMPONENT-BASED VERIFICATION

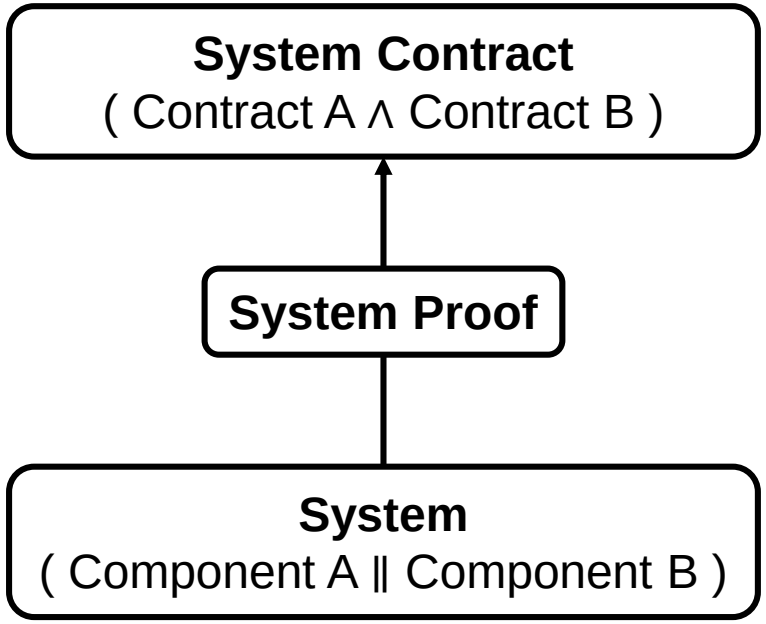
Monolithic

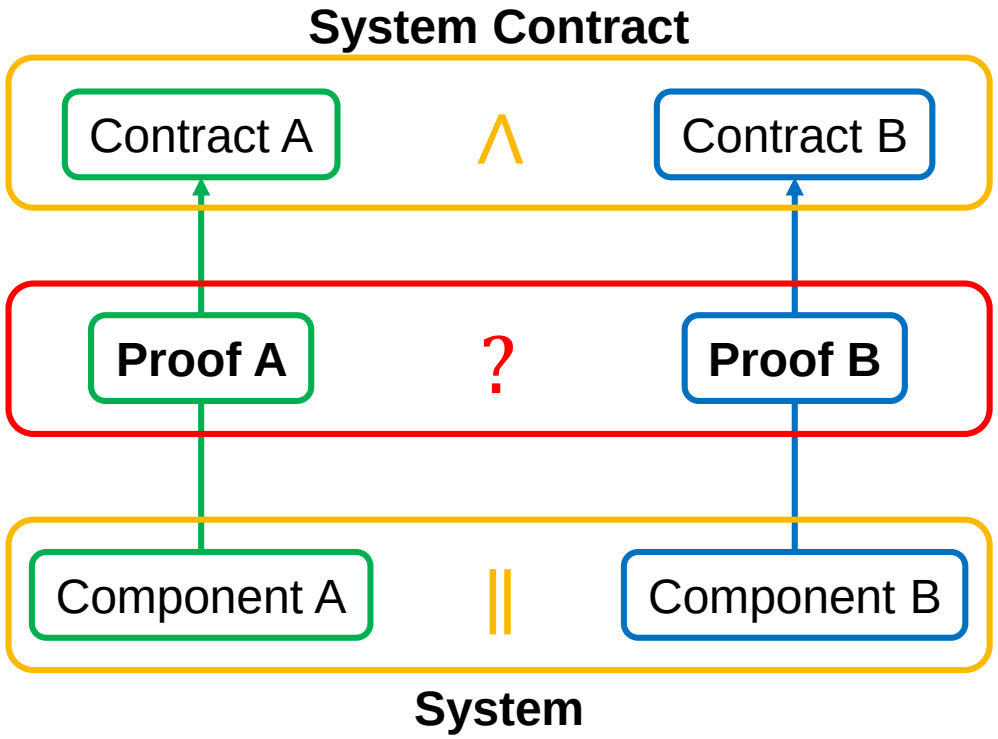
Component-based

External Behavior

Formal Verification

Internal Behavior





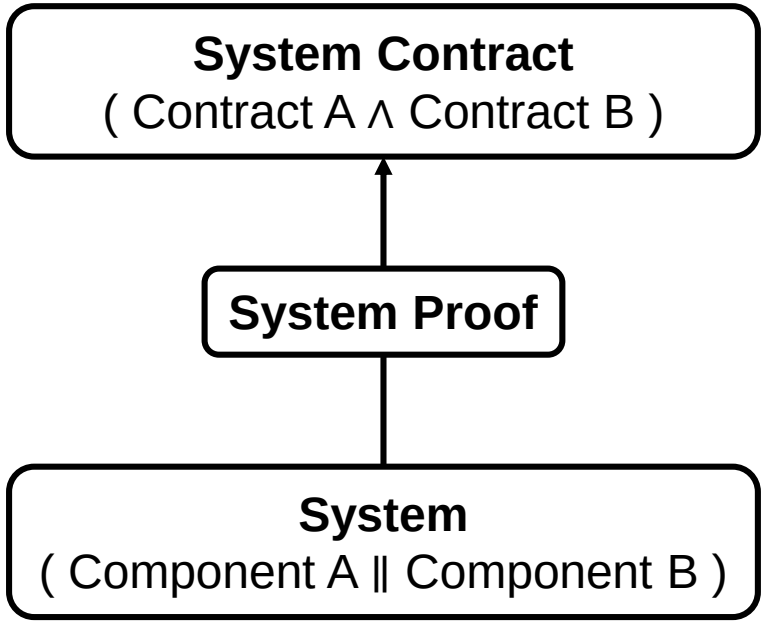
BACKGROUND – COMPONENT-BASED VERIFICATION

Monolithic

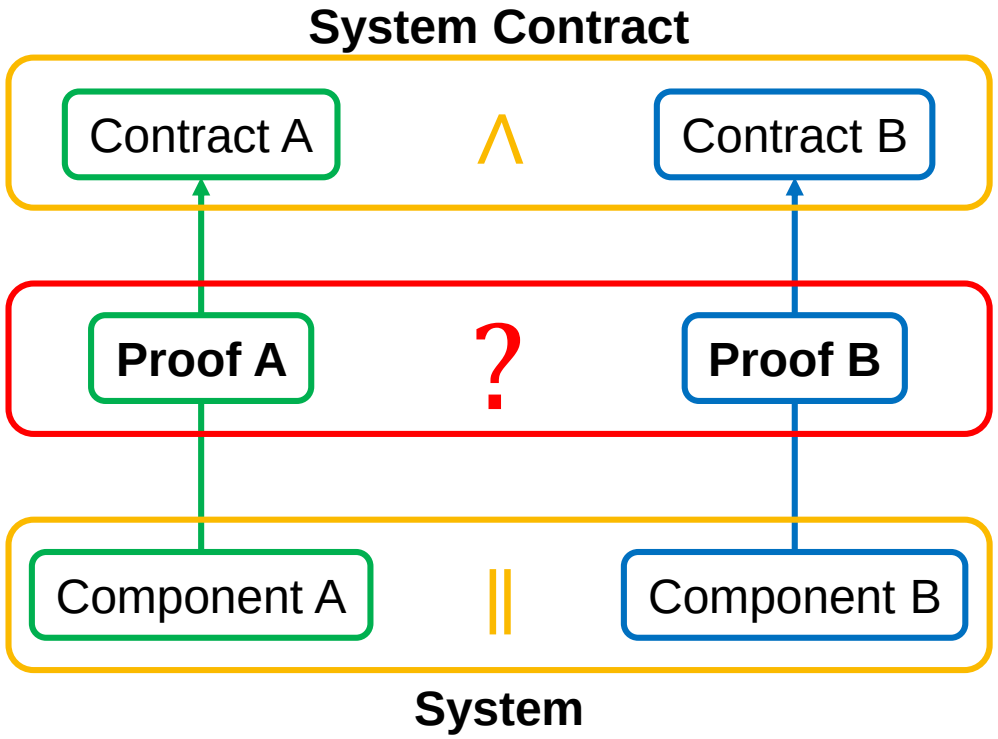
External Behavior

Formal Verification

Internal Behavior



Component-based



OVERVIEW

- Background
  - Cyber-Physical Systems
  - Component-based Verification

- Component-based Verification
  - Components
  - Contracts
  - Composition

- Evaluation
  - Implementation
  - Experiments

COMPONENT-BASED VERIFICATION – COMPONENTS

COMPONENT-BASED VERIFICATION – COMPONENTS

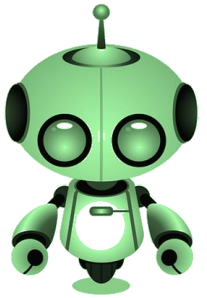
- **Components** arise naturally
  - Different parts with different responsibilities

COMPONENT-BASED VERIFICATION – COMPONENTS

- **Components** arise naturally
  - Different parts with different responsibilities

- Consist of discrete **control** part and a continuous **plant** part
  → describe component behavior

COMPONENT-BASED VERIFICATION – COMPONENTS

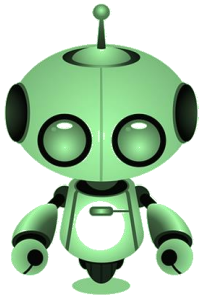


- **Components** arise naturally
  - Different parts with different responsibilities

- Consist of discrete **control** part and a continuous **plant** part
  → describe component behavior

- control ≈ if $(\boldsymbol{SAFE})$ $speed \coloneqq \boldsymbol{spAdv}$ else $speed \coloneqq 0$
- plant ≈ $pos'(t) = speed$

COMPONENT-BASED VERIFICATION – COMPONENTS



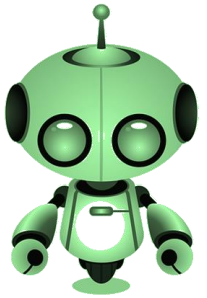
- **Components** arise naturally
  - Different parts with different responsibilities

- Consist of discrete **control** part and a continuous **plant** part
  → describe component behavior

- Read **input** values
  → allows component interaction

- control ≈ if $(\boldsymbol{SAFE})$ $speed \coloneqq \boldsymbol{spAdv}$ else $speed \coloneqq 0$
- plant ≈ $pos'(t) = speed$

COMPONENT-BASED VERIFICATION – COMPONENTS



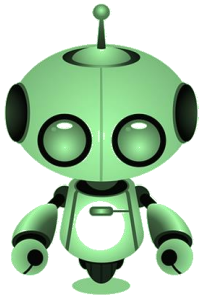
- **Components** arise naturally
  - Different parts with different responsibilities

- Consist of discrete **control** part and a continuous **plant** part
  → describe component behavior

- Read **input** values
  → allows component interaction

- control ≈ if $(\boldsymbol{SAFE})$ $speed \coloneqq \boldsymbol{spAdv}$ else $speed \coloneqq 0$
- plant ≈ $pos'(t) = speed$
- in ≈ $spAdv \coloneqq in_1$ $obsPos \coloneqq in_2$

COMPONENT-BASED VERIFICATION – COMPONENTS



- **Components** arise naturally
  - Different parts with different responsibilities

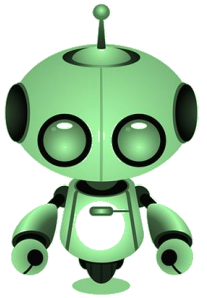
- Consist of discrete **control** part and a continuous **plant** part
  → describe component behavior

- Read **input** values
  → allows component interaction

- Repeatedly execute resulting program

- control ≈ if $(\boldsymbol{SAFE})$ $speed \coloneqq \boldsymbol{spAdv}$ else $speed \coloneqq 0$
- plant ≈ $pos'(t) = speed$
- in ≈ $spAdv \coloneqq in_1$ $obsPos \coloneqq in_2$

COMPONENT-BASED VERIFICATION – COMPONENTS



- **Components** arise naturally
  - Different parts with different responsibilities

- Consist of discrete **control** part and a continuous **plant** part
  → describe component behavior

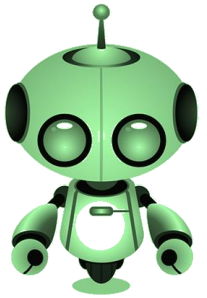
- Read **input** values
  → allows component interaction

- Repeatedly execute resulting program

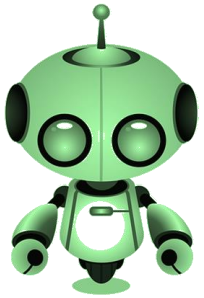
- control ≈ if $(\boldsymbol{SAFE})$ $speed \coloneqq \boldsymbol{spAdv}$ else $speed \coloneqq 0$
- plant ≈ $pos'(t) = speed$
- in ≈ $spAdv \coloneqq in_1$ $obsPos \coloneqq in_2$

$$\textbf{program} \approx (\ \boldsymbol{ctrl}\ ;\boldsymbol{plant}\ ;\boldsymbol{in}\ )*$$

COMPONENT-BASED VERIFICATION – CONTRACTS

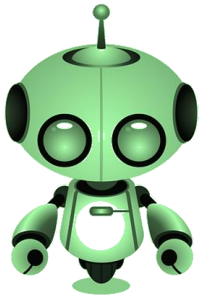


COMPONENT-BASED VERIFICATION – CONTRACTS



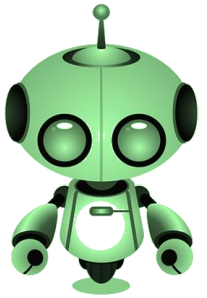
- Initial condition

COMPONENT-BASED VERIFICATION – CONTRACTS



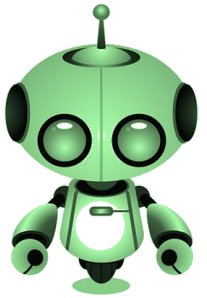
- **Initial condition**
  - ☐ Choose validity ranges for **global constants**

COMPONENT-BASED VERIFICATION – CONTRACTS



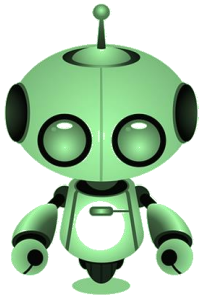
- **Initial condition**
  - Choose validity ranges for **global constants**
- initial ≈
  - Globally constant parameters, e.g., $speed_{max} > 0$

COMPONENT-BASED VERIFICATION – CONTRACTS



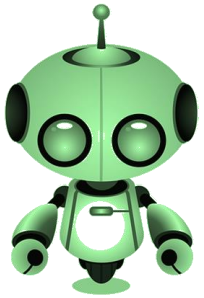
- **Initial condition**
  - Choose validity ranges for **global constants**
  - Additional **assumptions** regarding initial state
- initial ≈
  - Globally constant parameters, e.g., $speed_{max} > 0$

COMPONENT-BASED VERIFICATION – CONTRACTS



- **Initial condition**
  - ☐ Choose validity ranges for **global constants**
  - ☐ Additional **assumptions** regarding initial state
- initial ≈
  - ☐ Globally constant parameters, e.g., $speed_{max} > 0$
  - ☐ Assumptions, e.g., $speed = 0$

COMPONENT-BASED VERIFICATION – CONTRACTS

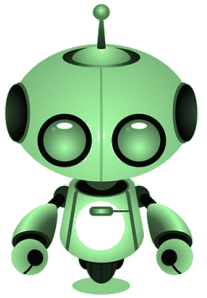


- **Initial condition**
  - Choose validity ranges for **global constants**
  - Additional **assumptions** regarding initial state

- **Safety condition**
  - Describes **safety** property and (optional) **guarantees** for values produced on **ports**

- initial ≈
  - Globally constant parameters, e.g., $speed_{max} > 0$
  - Assumptions, e.g., $speed = 0$

COMPONENT-BASED VERIFICATION – CONTRACTS



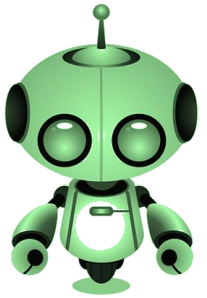
- **Initial condition**
  - Choose validity ranges for **global constants**
  - Additional **assumptions** regarding initial state

- **Safety condition**
  - Describes **safety** property and (optional) **guarantees** for values produced on **ports**

- initial ≈
  - Globally constant parameters, e.g., $speed_{max} > 0$
  - Assumptions, e.g., $speed = 0$

- post ≈
  - Guarantees, e.g., $speed > 0 \rightarrow robPos \neq obsPos$

COMPONENT-BASED VERIFICATION – CONTRACTS



- **Initial condition**
  - Choose validity ranges for **global constants**
  - Additional **assumptions** regarding initial state
- **Safety condition**
  - Describes **safety** property and (optional) **guarantees** for values produced on **ports**

- initial ≈
  - Globally constant parameters, e.g., $speed_{max} > 0$
  - Assumptions, e.g., $speed = 0$
- post ≈
  - Guarantees, e.g., $speed > 0 \rightarrow robPos \neq obsPos$

$$\textbf{contract} \approx \boldsymbol{initial} \rightarrow [\ \textbf{program}\ ]\ \boldsymbol{post}$$

COMPONENT-BASED VERIFICATION – COMPOSITION (1)

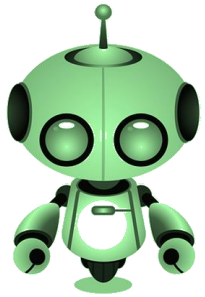
COMPONENT-BASED VERIFICATION – COMPOSITION (1)

- Parallel Composition

COMPONENT-BASED VERIFICATION – COMPOSITION (1)

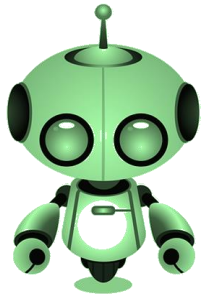
- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel

COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time

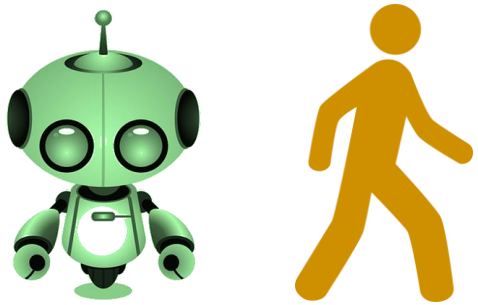
COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown

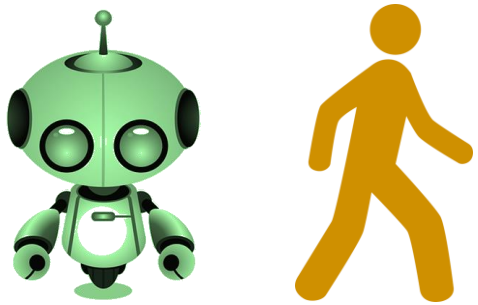
- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time

COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown
- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time
  - Compose control part
    - Robot chooses speed and direction
    - Obstacle chooses speed and direction

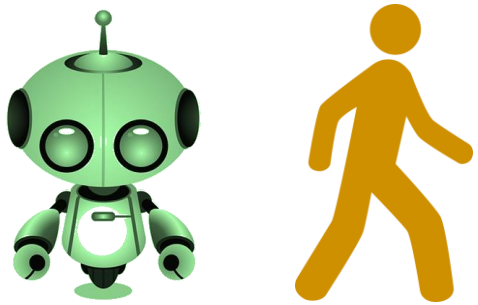
COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown
    - → Non-deterministically chosen sequence of control parts

- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time
  - Compose control part
    - Robot chooses speed and direction
    - Obstacle chooses speed and direction

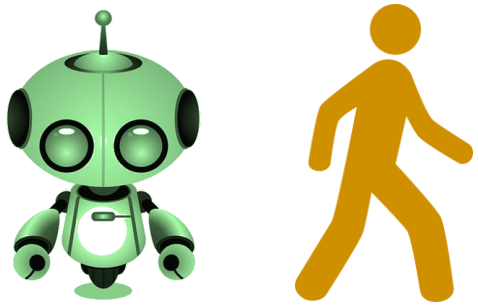
COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown
    - → Non-deterministically chosen sequence of control parts

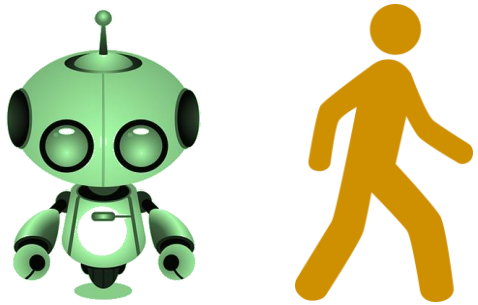
- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time
  - Compose control part
    - Robot chooses speed and direction
    - Obstacle chooses speed and direction
    - Robot, Obstacle OR Obstacle, Robot

COMPONENT-BASED VERIFICATION – COMPOSITION (1)



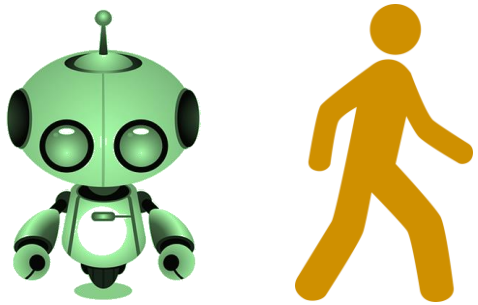
- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown
    - → Non-deterministically chosen sequence of control parts
    - Alternative: Expert system (select most-reasonable control sequence)
- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time
  - Compose control part
    - Robot chooses speed and direction
    - Obstacle chooses speed and direction
    - Robot, Obstacle OR Obstacle, Robot

COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown
    - → Non-deterministically chosen sequence of control parts
    - Alternative: Expert system (select most-reasonable control sequence)
- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time
  - Compose control part
    - Robot chooses speed and direction
    - Obstacle chooses speed and direction
    - Robot, Obstacle OR Obstacle, Robot
    - Expert: Robot always chooses first → Robot, Obstacle

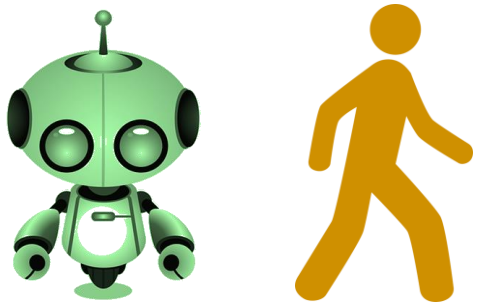
COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown
    - → Non-deterministically chosen sequence of control parts
    - Alternative: Expert system (select most-reasonable control sequence)
  - Compatible Communication: Only compatible ports can be connected

- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time
  - Compose control part
    - Robot chooses speed and direction
    - Obstacle chooses speed and direction
    - Robot, Obstacle OR Obstacle, Robot
    - Expert: Robot always chooses first → Robot, Obstacle

COMPONENT-BASED VERIFICATION – COMPOSITION (1)



- Parallel Composition
  - Compose plant part
    - Time passes simultaneously
    - → Plants must be executed in parallel
  - Compose control part
    - Assumption: Control takes no time
    - Execution order is crucial, but unknown
    - → Non-deterministically chosen sequence of control parts
    - Alternative: Expert system (select most-reasonable control sequence)
  - Compatible Communication: Only compatible ports can be connected

- e.g., Robot and Obstacle
  - Compose plant part
    - Robot and obstacle move at the same time
  - Compose control part
    - Robot chooses speed and direction
    - Obstacle chooses speed and direction
    - Robot, Obstacle OR Obstacle, Robot
    - Expert: Robot always chooses first → Robot, Obstacle
  - Compatible Communication
    - Robot expects position of obstacle close to previous position

COMPONENT-BASED VERIFICATION – COMPOSITION (2)

- Composite program

$$prog_3 \approx \Big(\; (ctrl_1; ctrl_2 \cup ctrl_2; ctrl_1) \; ; \; (plant_1 \parallel plant_2) \; ; in_1; in_2; in_{open} \; \Big) *$$

- Composite contract

$$cont_3 \approx (\; init_1 \wedge init_2 \;) \; \rightarrow [\; prog_3 \;] \; (\; post_1 \wedge post_2 \;)$$

- **Theorem**
  - ☐ Composite program of two compatible components obeys composite contract
  - ☐ User provides **Proof A** and **Proof B**
  - ☐ Theorem derives **System Proof**

→ Safety verification results about contracts for components transfer to composites!

COMPONENT-BASED VERIFICATION – COMPOSITION (2)

- Composite program

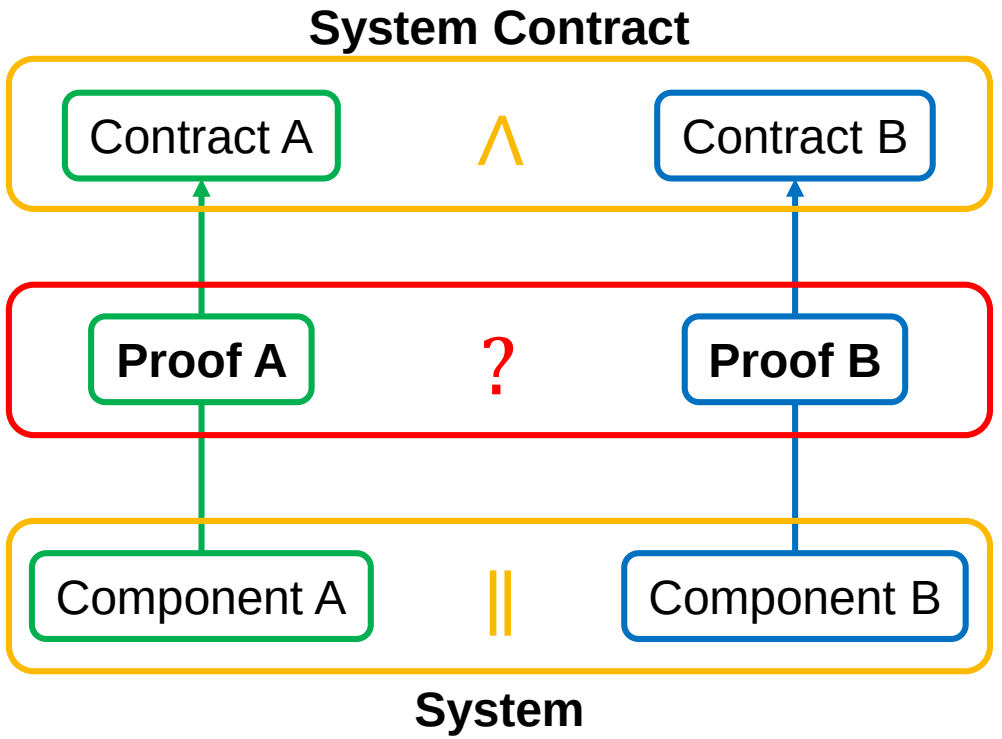
$$prog_3 \approx \left(\; (ctrl_1; ctrl_2 \cup ctrl_2; ctrl_1) \; ; \; (plant_1 \parallel plant_2) \; ; in_1; in_2; in_{open} \; \right) *$$

- Composite contract

$$cont_3 \approx (\; init_1 \wedge init_2 \;) \; \rightarrow [\; prog_3 \;] \; (\; post_1 \wedge post_2 \;)$$

- **Theorem**
  - Composite program of two compatible components obeys composite contract
  - User provides **Proof A** and **Proof B**
  - Theorem derives **System Proof**

→ Safety verification results about contracts for components transfer to composites!



COMPONENT-BASED VERIFICATION – COMPOSITION (2)

- Composite program

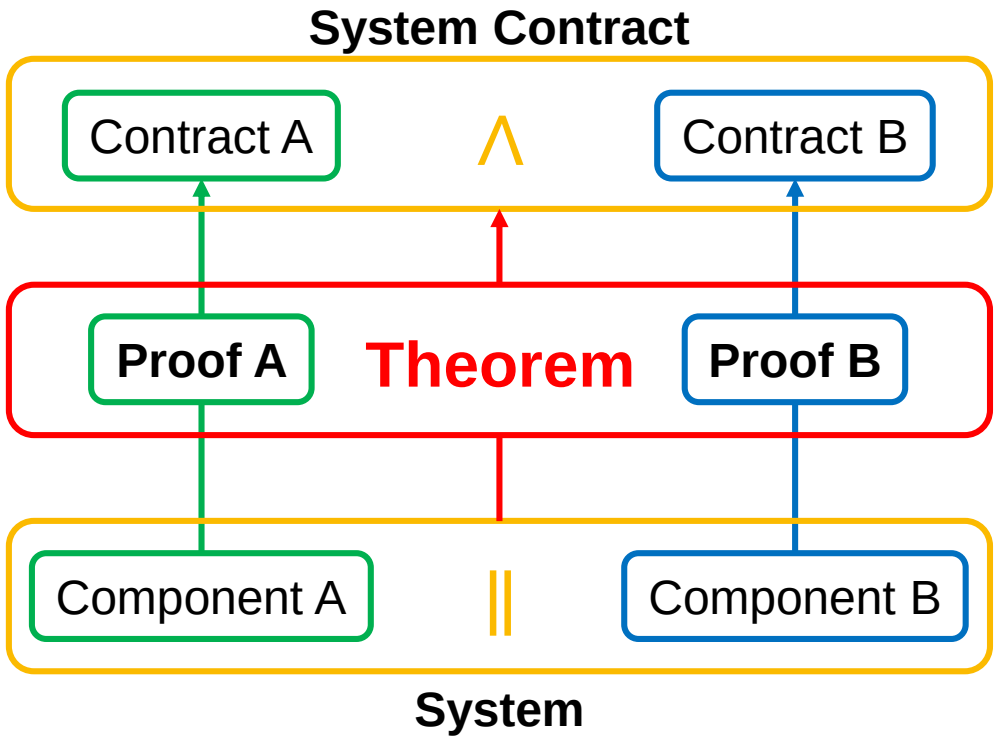
$$prog_3 \approx \left(\; (ctrl_1; ctrl_2 \cup ctrl_2; ctrl_1) \; ; \; (plant_1 \parallel plant_2) \; ; in_1; in_2; in_{open} \; \right) *$$

- Composite contract

$$cont_3 \approx (\; init_1 \wedge init_2 \;) \; \rightarrow [\; prog_3 \;] \; (\; post_1 \wedge post_2 \;)$$

- **Theorem**
  - Composite program of two compatible components obeys composite contract
  - User provides **Proof A** and **Proof B**
  - Theorem derives **System Proof**

→ Safety verification results about contracts for components transfer to composites!



OVERVIEW

- Background
  - Cyber-Physical Systems
  - Component-based Verification

- Component-based Verification
  - Components
  - Contracts
  - Composition

- Evaluation
  - Implementation
  - Experiments

EVALUATION – IMPLEMENTATION

- KeYmaera X
  - Provides built-in proof rules to transform models
  - Tactic: combination of proof rules that transforms model to something known

EVALUATION – IMPLEMENTATION

- KeYmaera X
  - Provides built-in proof rules to transform models
  - Tactic: combination of proof rules that transforms model to something known
- Implementation of theorem as additional proof rule?

EVALUATION – IMPLEMENTATION

- KeYmaera X
  - Provides built-in proof rules to transform models
  - Tactic: combination of proof rules that transforms model to something known
- Implementation of theorem as additional proof rule?

EVALUATION – IMPLEMENTATION

- KeYmaera X
  - Provides built-in proof rules to transform models
  - Tactic: combination of proof rules that transforms model to something known
- Implementation of theorem as additional proof rule?

- Derive tactic that verifies system based on component tactics
  - No safety critical changes necessary

EVALUATION – IMPLEMENTATION

- KeYmaera X
  - Provides built-in proof rules to transform models
  - Tactic: combination of proof rules that transforms model to something known
- Implementation of theorem as additional proof rule?
- Derive tactic that verifies system based on component tactics
  - No safety critical changes necessary

EVALUATION – IMPLEMENTATION

- KeYmaera X
  - Provides built-in proof rules to transform models
  - Tactic: combination of proof rules that transforms model to something known
- Implementation of theorem as additional proof rule?

- Derive tactic that verifies system based on component tactics
  - No safety critical changes necessary

- Input
  - Components, i.e., control and plant
  - Contract, i.e., pre- and post condition
  - Tactic to proof that components obey respective contracts
- Output
  - Composite program
  - Composite contract
  - Tactic to proof that composite program obeys composite contract

EVALUATION – IMPLEMENTATION

- KeYmaera X
  - Provides built-in proof rules to transform models
  - Tactic: combination of proof rules that transforms model to something known
- Implementation of theorem as additional proof rule?

- Derive tactic that verifies system based on component tactics
  - No safety critical changes necessary

- Input
  - Components, i.e., control and plant
  - Contract, i.e., pre- and post condition
  - Tactic to proof that components obey respective contracts
- Output
  - Composite program
  - Composite contract
  - Tactic to proof that composite program obeys composite contract

Safe Component + Safe Component = Safe System

EVALUATION – EXPERIMENTS

- Existing Case Studies
  - Robot Collision Avoidance – Robix
  - European Train Control System – ETCS
  - Adaptive Cruise Control – LLC
- Robot Collision Avoidance – RC
- Verified monolithic and component-based

EVALUATION – EXPERIMENTS

- Existing Case Studies
  - Robot Collision Avoidance – Robix
  - European Train Control System – ETCS
  - Adaptive Cruise Control – LLC
- Robot Collision Avoidance – RC
- Verified monolithic and component-based

	Non-linear	Manual steps		Duration [s]	
		Comp	Mono	Comp	Mono
ETCS		0	0	873	15306
Robix	x	31	96	718	902
LLC		50	131	753	568
RC		0	0	189	1934

EVALUATION – EXPERIMENTS

- Existing Case Studies
  - Robot Collision Avoidance – Robix
  - European Train Control System – ETCS
  - Adaptive Cruise Control – LLC
- Robot Collision Avoidance – RC
- Verified monolithic and component-based

	Non-linear	Manual steps		Duration [s]	
		Comp	Mono	Comp	Mono
ETCS		0	0	873	15306
Robix	x	31	96	718	902
LLC		50	131	753	568
RC		0	0	189	1934

- Summary
  - Reduction of verification time (especially for automated proofs)
  - Reduction of proof effort

Safe Component + Safe Component
=
Safe System

CHANGE AND DELAY CONTRACTS FOR HYBRID SYSTEM COMPONENT VERIFICATION

Andreas Müller – andreas.mueller@jku.at
Werner Retschitzegger – werner.retschitzegger@jku.at
Wieland Schwinger – wieland.schwinger@jku.at

Johannes Kepler University, Linz
Department of Cooperative Information Systems
http://cis.jku.at/

Stefan Mitsch – smitsch@cs.cmu.edu
André Platzer - aplatzer@cs.cmu.edu

Carnegie Mellon University, Pittsburgh
Computer Science Department
http://www.ls.cs.cmu.edu