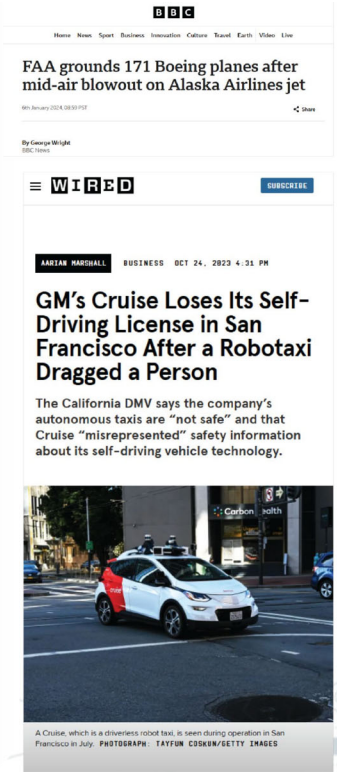
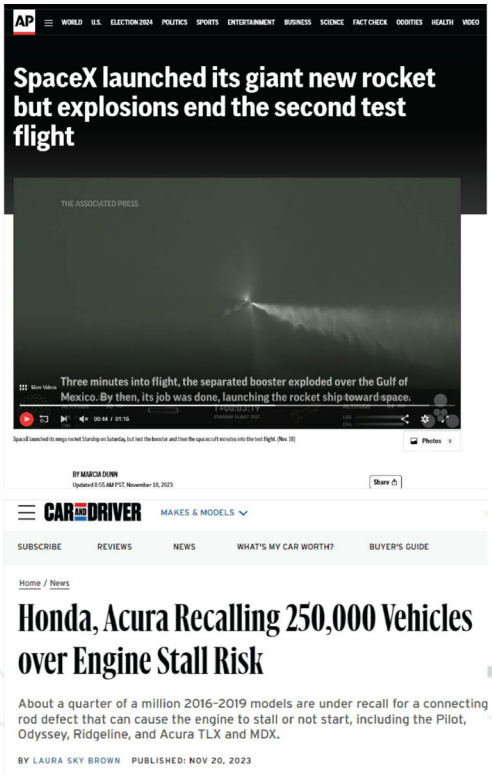
Towards Provably Safe and Secure Systems with Contract-Based Design

Embedded Systems Seminar
19 Sept 2024

Inigo Incer
iir@umich.edu
https://iincer.github.io

Electrical and Computer Engineering
University of Michigan





Improve how the Agency Engineering Domain operates over the entire NASA lifecycle by effectively managing complexity, reducing cost and schedule, and improving product integrity via ***the integration of processes, digital tools, and techniques along with seamless flow of information throughout the engineering system development life-cycle*** *(concept development, design, testing and validation, manufacturing and operations)*

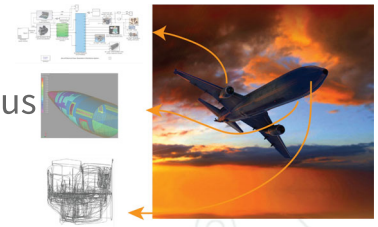
— Terry Hill (NASA DT Digital Engineering Lead)

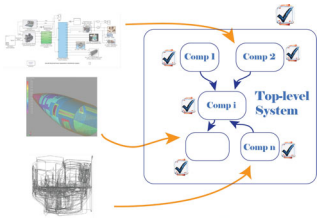
Challenges in complex system design

Multiple stakeholders

Complex supply chains

Heterogenous analyses





System integration

Contracts

An algebra and a methodology to address

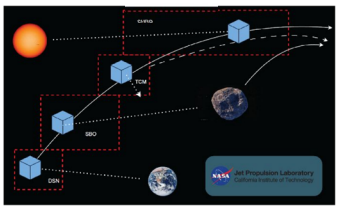
- System integration
  - Subsystems come from multiple OEMs
  - Integrate through **composition**
  - Find “missing components” through **quotient**
- Multiple viewpoints
  - At the same company, multiple groups work on same product
  - Merge them through **merging**

Albert Benveniste et al. Contracts for System Design. Foundations and Trends in Electronic Design Automation 2018
Inigo Incer. The Algebra of Contracts, PhD thesis, UC Berkeley 2022

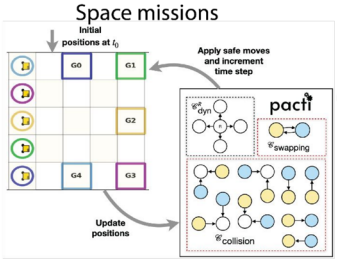
pacti .ORG

Inigo Incer et al.: Pacti: Scaling assume-guarantee reasoning for system analysis and design. arXiv:2303.17751 2023. Under review.

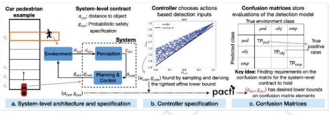
CPS



Space missions

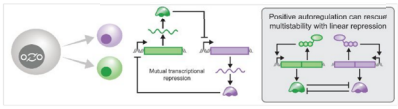


Multi-agent planning

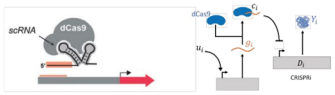


Perception in autonomous vehicles

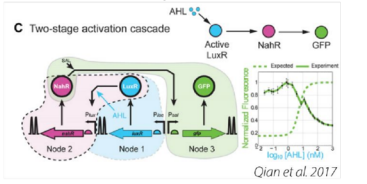
Synthetic Biology



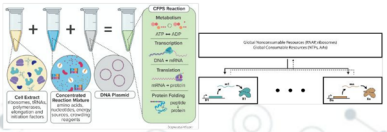
Mammalian cell toggle-switch



Resource competition in CRISPR



Sequential signal processing in bacteria



Cell-free biosensing

pacti .ORG

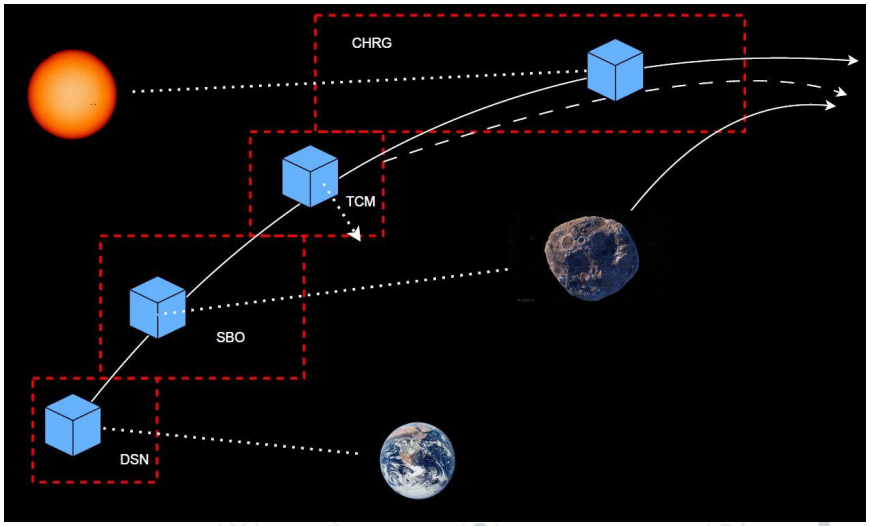
Outline

1. NASA JPL's space-mission case study
2. Quantifying the robustness of perception components
3. Theory of AG contracts
4. Conclusions

Outline

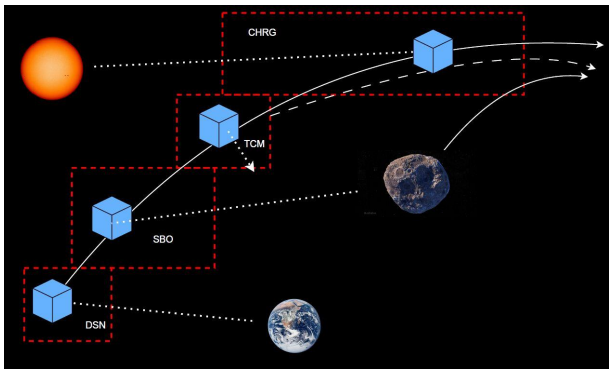
1. NASA JPL's space-mission case study
2. Quantifying the robustness of perception components
3. Theory of AG contracts
4. Conclusions

Designing a space mission



Nicolas Rouquette, **Inigo Incer**, Alessandro Pinto. Early Design Exploration of Space System Scenarios Using Assume-Guarantee Contracts. IEEE SMC-IT 2023.

Designing a spacecraft for a mission



Given n repetitions of the 4 scenario tasks:

DSN_1, SBO_1, TCM_1, $CHRG_1$, …, $CHRG_n$, TCM_n, $CHRG_{n+1}$

Q1. Can the spacecraft maintain adequate power throughout?

Q2. Can the spacecraft record and transmit all asteroid observations?

Current design processes

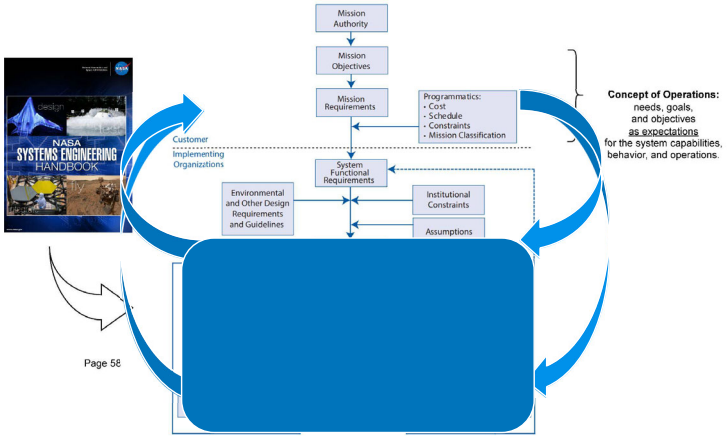


FIGURE 4.2-3 The Flowdown of Requirements

https://www.nasa.gov/connect/ebooks/nasa-systems-engineering-handbook

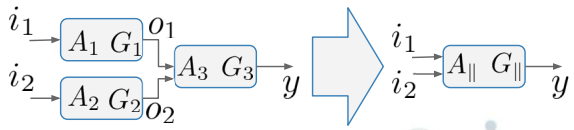
Assume-guarantee contracts

- We will represent all components in a system using ***assume-guarantee contracts***
- Contracts are formal specifications stating
  - (i) what a component does and
  - (ii) what it expects from its environment in order to deliver its guarantees
- Contracts have notion of order
- They have many system-relevant operations

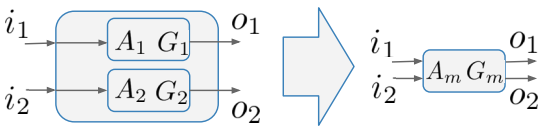
Contract = (assumptions, guarantees)

Eg:
(at most 30% of location sensors are attacked, the self-driving car reaches its goal with at least 99% chance)

Building systems

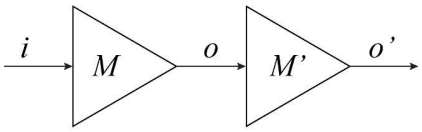


Merging viewpoints



Albert Benveniste et al. Contracts for System Design. Foundations and Trends in Electronic Design Automation 2018
Inigo Incer et al.: Pacti: Scaling assume-guarantee reasoning for system analysis and design. arXiv:2303.17751 2023. Under review.
Roberto Passerone, **Inigo Incer,** Alberto L. Sangiovanni-Vincentelli: Coherent Extension, Composition, and Merging Operators in Contract Models for System Design. ACM TECS 2019.

Composition example



$$\mathcal{C} = (|i| \leq 2, o = i) \qquad \mathcal{C}' = \left(|o| \leq 1, o' = o\right)$$

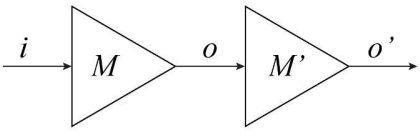
$$\mathcal{C}_S = \left(|i| \leq 1, o' = i\right)$$

```python
contract1 = PolyhedralContract.from_string(
    input_vars=["i"],
    output_vars=["o"],
    assumptions=["|i| <= 2"],
    guarantees=["o - i = 0"]
)

contract2 = PolyhedralContract.from_string(
    input_vars=["o"],
    output_vars=["o_p"],
    assumptions=["|o| <= 1"],
    guarantees=["o_p - o = 0"]
)

system_contract = contract1.compose(contract2)
print(system_contract)
```

✓ 0.1s

```
InVars: [i]
OutVars:[o_p]
A: [
  |i| <= 1
]
G: [
  -i + o_p = 0
]
```


Diagnostics example



$$\mathcal{C} = (|i| \leq 2, |o| \leq 3) \qquad \mathcal{C}' = \left(|o| \leq 1, o' = o\right)$$

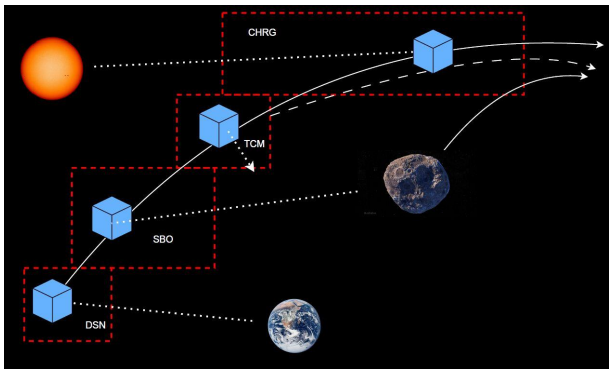
System has an interconnection problem that simulation is not guaranteed to catch

```python
contract1 = PolyhedralIoContract.from_strings(
    input_vars=["i"],
    output_vars=["o"],
    assumptions=["|i| <= 2"],
    guarantees=["|o| <= 3"]
)

contract2 = PolyhedralIoContract.from_strings(
    input_vars=["o"],
    output_vars=["o_p"],
    assumptions=["|o| <= 1"],
    guarantees=["o_p = o"]
)

try:
    system_contract = contract1.compose(contract2)
except ValueError as e:
    print("Composition error: {0}".format(e))
```

✓ 0.0s

```
Composition error: Could not eliminate variables ['o', 'o_p']
by refining the assumptions
[
  |o| <= 1
]
using guarantees
[
  |i| <= 2
  |o| <= 3
]
```

Designing a spacecraft for a mission I



Given n repetitions of the 4 scenario tasks:

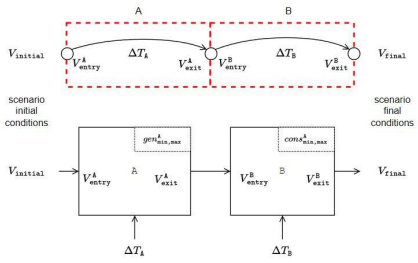
DSN_1, SBO_1, TCM_1, $CHRG_1$, …, $CHRG_n$, TCM_n, $CHRG_{n+1}$

Q1. Can the spacecraft maintain adequate **power** throughout?

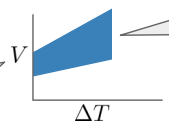
Q2. Can the spacecraft record and transmit all **asteroid observations**?

Designing a spacecraft for a mission II

- The questions generate state variables:
  - state of charge
  - scientific data gathered
  - data transmitted to earth
  - relative distance to SBO
- Each mode of operation makes guarantees on each state (viewpoint)



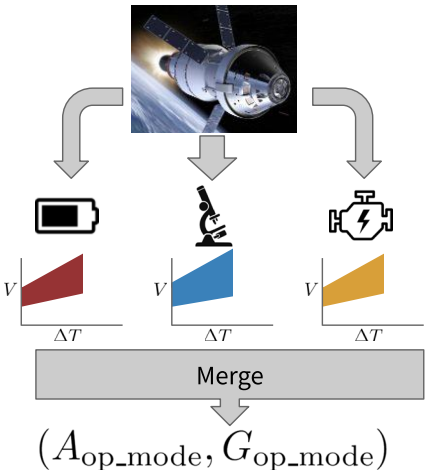
Assumptions: acceptable initial values for state



Guarantees: range of output state values (time-dependent)

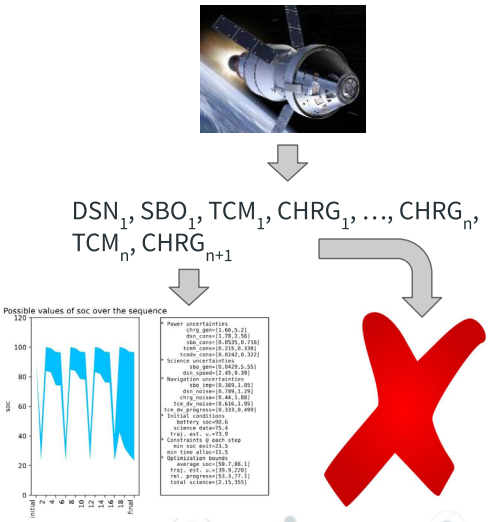
Designing a spacecraft for a mission III

- Each spacecraft design has a contract for each viewpoint (power, distance to objective, etc.)
- To get the contract of every mode of operation, we merge its viewpoints
  - Each mode of operation makes guarantees of each state
  - And makes assumptions on initial state



Designing a spacecraft for a mission IV

- For each spacecraft, we can ask whether the mission can be accomplished by composing the contracts in the schedule
- If the composition of contracts is successful, the spacecraft can complete the mission



Efficiently designing a spacecraft for a mission

- We considered 200 spacecraft configurations and schedules of 5 and 20 steps

- The analysis for each spacecraft was recomputed by varying the top-level requirements
  - 200 configurations and 100 variations of requirements

Scenario	Pacti operations for scenario generation
5-step	compose: 2016, up to 22 constraints, 12 variables merge: 1680, up to 44 constraints, 23 variables total time: 16.3 seconds
20-step	compose: 10,200, up to 187 constraints, 95 variables merge: 8,000, up to 44 constraints, 23 variables total time: 97.8 seconds

Scenario	Pacti operations for schedulability analysis
5-step	merge: 181,988, up to 81 constraints, 35 variables admissible solutions: 401 out of 20,000 combinations total time: 3 minutes, 9.6 seconds
20-step	merge: 781,331, up to 275 constraints, 125 variables admissible solutions: 244 out of 20,000 combinations total time: 4 minutes, 11 seconds

Current design processes

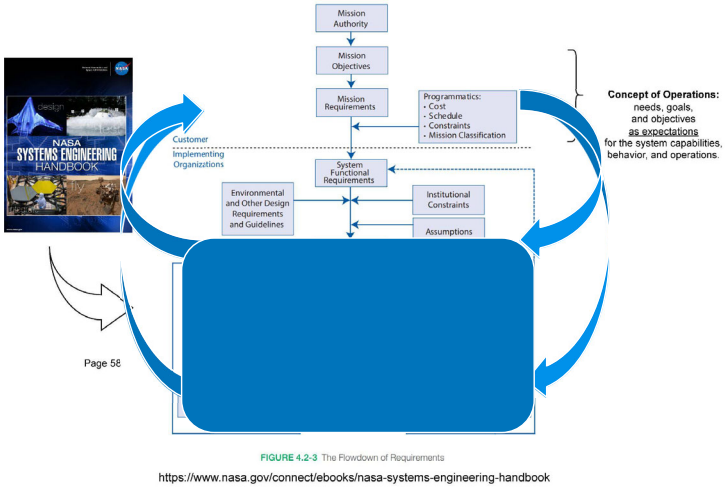
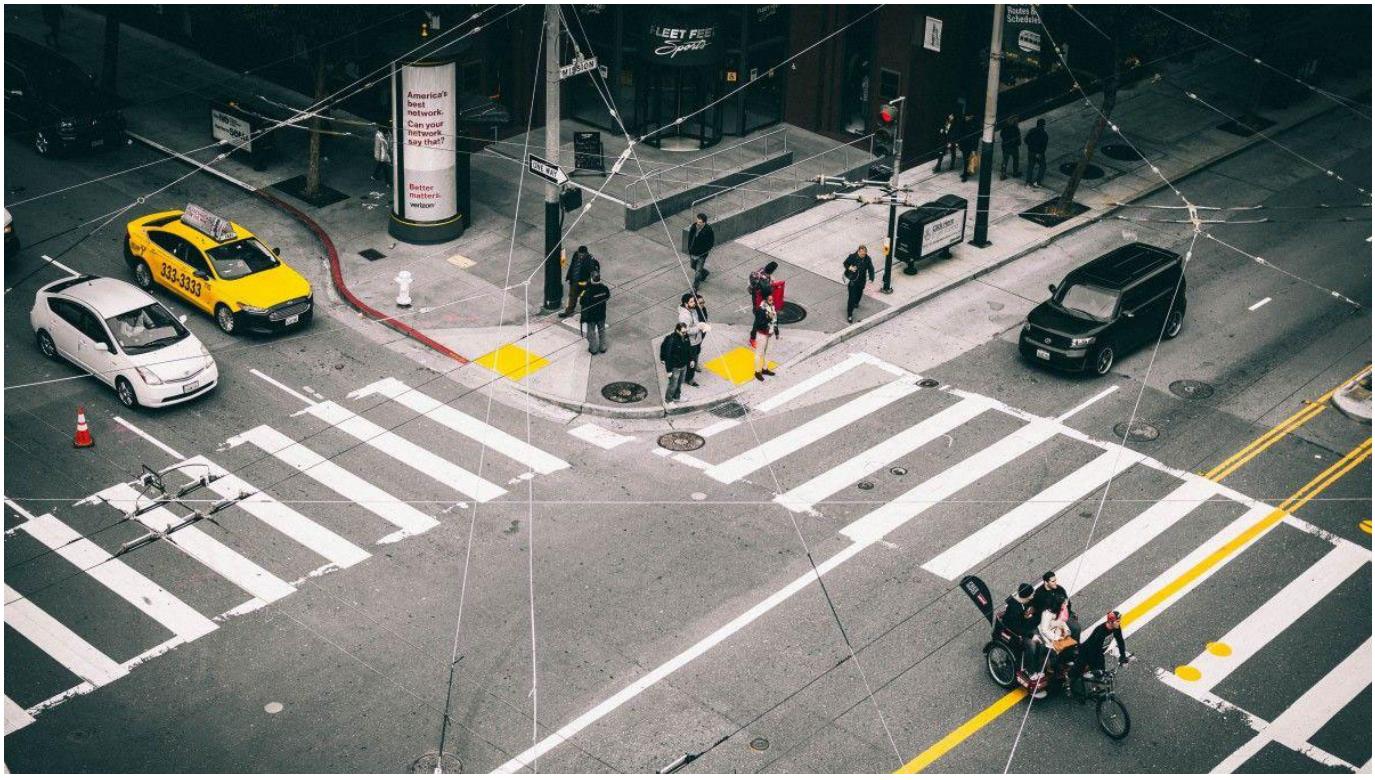


FIGURE 4.2-3 The Flowdown of Requirements

https://www.nasa.gov/connect/ebooks/nasa-systems-engineering-handbook

Outline

1. NASA JPL's space-mission case study
2. Quantifying the robustness of perception components
3. Theory of AG contracts
4. Conclusions

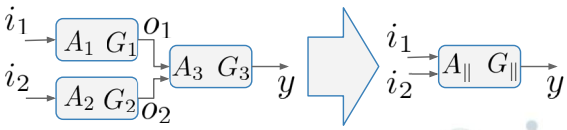


Assume-guarantee contracts

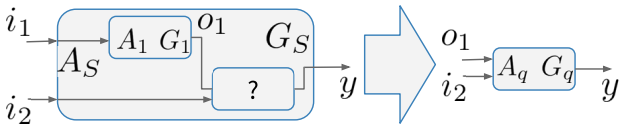
- We will represent all components in a system using ***assume-guarantee contracts***
- Contracts are formal specifications stating
  - (i) what a component does and
  - (ii) what it expects from its environment in order to deliver its guarantees
- They have many system-relevant operations:

$$\mathcal{C} = (A, G)$$

Building systems

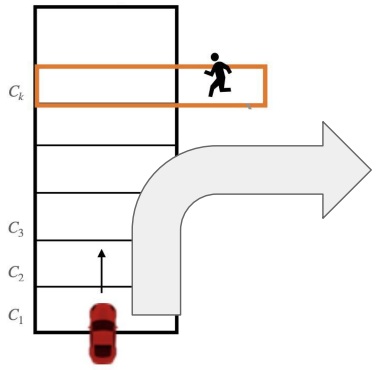


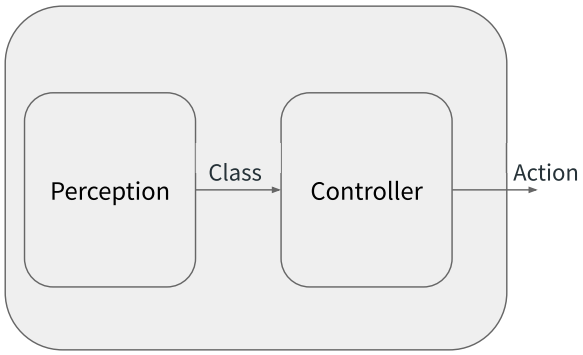
Decomposing systems



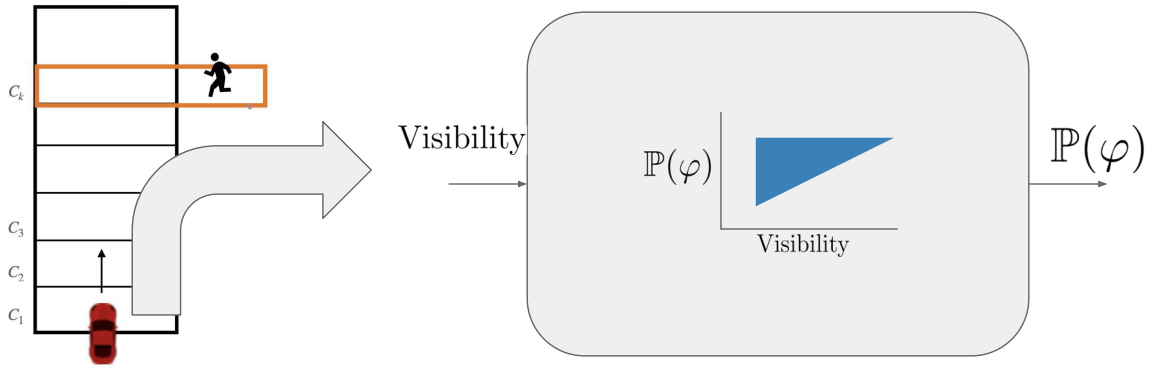
Inigo Incer, Alberto Sangiovanni-Vincentelli, Chung-Wei Lin, Eunsuk Kang. Quotient for Assume-Guarantee Contracts. MEMOCODE 2018.

Deriving perception requirements

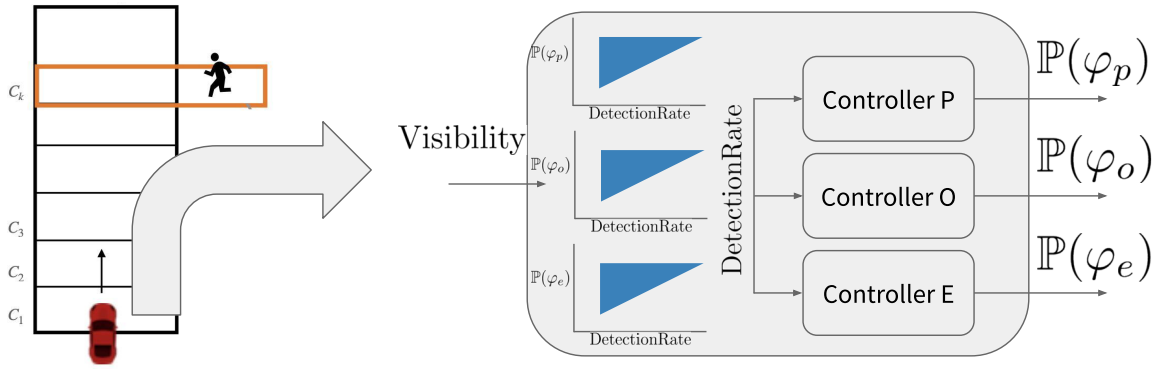




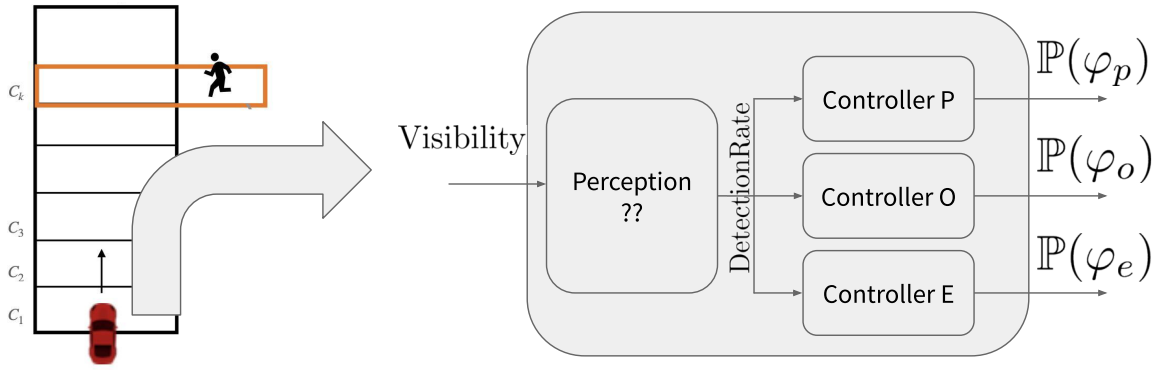
Deriving perception requirements



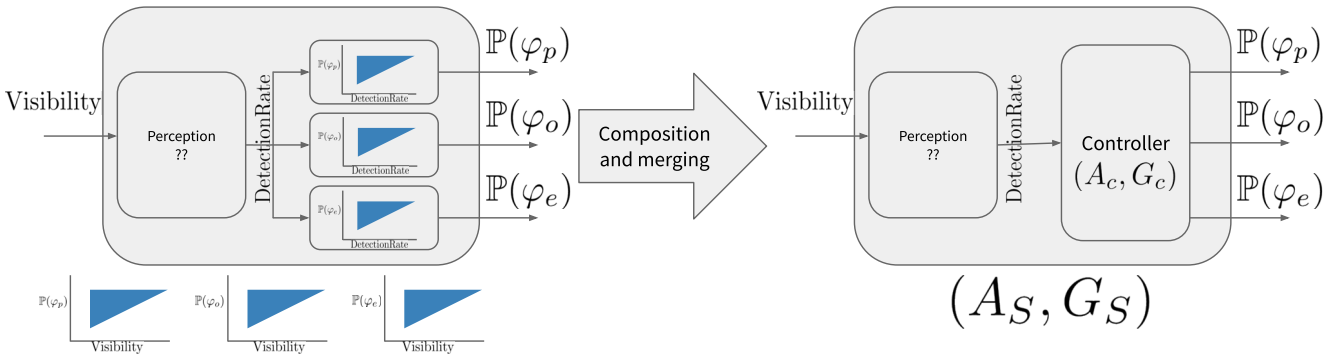
Deriving perception requirements



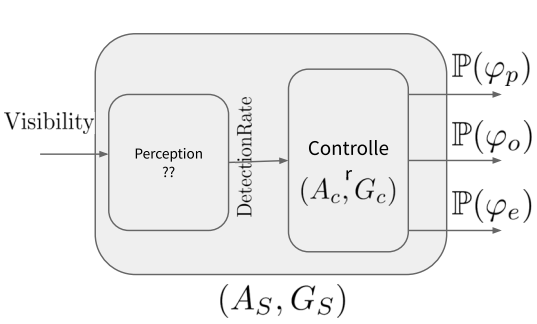
Deriving perception requirements



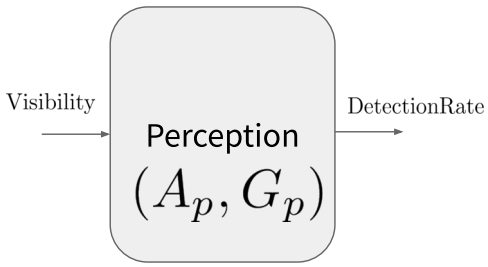
Deriving perception requirements

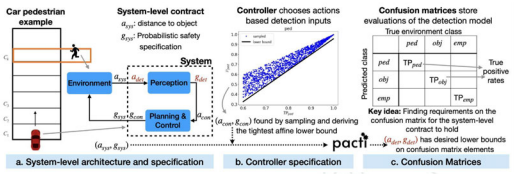


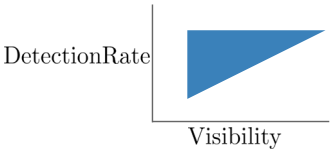
Deriving perception requirements



Quotient







Outline

1. NASA JPL's space-mission case study
2. Quantifying the robustness of perception components
3. Theory of AG contracts
4. Conclusions

The algebra of AG contracts

- Given a Boolean algebra ***B*** of specifications
- Its contract algebra is the set

Constraint ensures that guarantees are SAT when assumptions are UNSAT

$$\mathbb{C}(B) = \{(a, g) \in B^{\mathsf{op}} \times B \mid a \vee g = 1_B\}$$

- Contracts have the following order

$$(a, g) \leq (a', g') \text{ iff } (g \leq g') \wedge (a' \leq a)$$

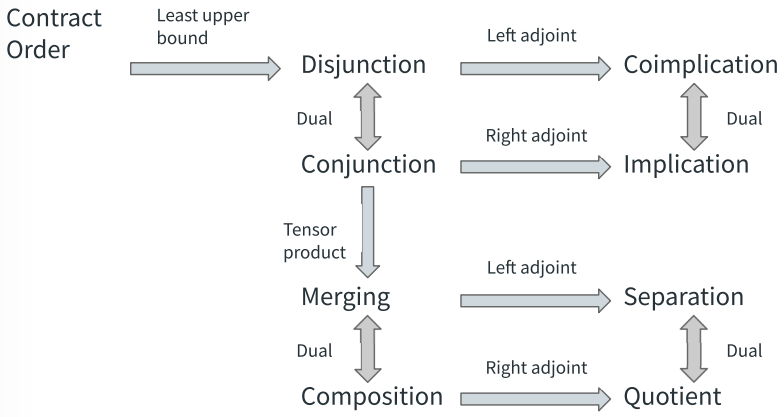
- The unary operation of reciprocal

$$(a, g)^{-1} = (g, a)$$

- Enables the definition of duality of operations

$$(\mathcal{C} \circ \mathcal{C}')^{-1} = \mathcal{C}^{-1} \diamond (\mathcal{C}')^{-1}$$

Contract operations I



Contract operations II

Conjunction $\mathcal{C} \wedge \mathcal{C}' = (a \vee a', g \wedge g')$	Disjunction $\mathcal{C} \vee \mathcal{C}' = (a \wedge a', g \vee g')$
Composition $\mathcal{C}_1 \parallel \mathcal{C}_2 = (a_1 \wedge a_2 \vee \neg(g_1 \wedge g_2), g_1 \wedge g_2)$	Merging $\mathcal{C}_1 \bullet \mathcal{C}_2 = (a_1 \wedge a_2, g_1 \wedge g_2 \vee \neg(a_1 \wedge a_2))$
Quotient $\mathcal{C}/\mathcal{C}' = (a \wedge g', g \wedge a' \vee \neg(a \wedge g'))$	Separation $\mathcal{C} \div \mathcal{C}' = (a \wedge g' \vee \neg(g \wedge a'), g \wedge a')$
Implication $\mathcal{C}' \rightarrow \mathcal{C} = ((a \wedge \neg a') \vee (g' \wedge \neg g), g \vee \neg g')$	Coimplication $\mathcal{C}' \nrightarrow \mathcal{C} = (a \vee \neg a', (g \wedge \neg g') \vee (a' \wedge \neg a))$

Inigo Incer: The Algebra of Contracts, PhD thesis, UC Berkeley, 2022.

Outline

1. NASA JPL's space-mission case study
2. Quantifying the robustness of perception components
3. Theory of AG contracts
4. Conclusions

What do we gain with contracts?

- We think of systems and subsystems in terms of their specifications—not implementation details
  - Conclusions will hold for any implementation adhering to its specification
- Contracts make assumptions first-class elements—we have to explicitly state the assumptions of all components
  - Operating on assumptions is less intuitive than on guarantees
- AG contracts enable us to scale compositional reasoning to large designs
  - The algebra has several design-relevant operations
- Contracts add logical AG reasoning to our system-level analysis; they don't replace our specialized tools
  - Contracts use control-theoretic tools to reason about control specifications, security tools to reason about security specifications, etc.