Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Institute for AI and Autonomous Systems

Conceptual Constraints

Uses for Cognitive Robots

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure

- What are conceptual constraints?
- An overview of applications of conceptual constraints
- Property-based robot testing

What are Conceptual Constraints?

Conceptual Constraints

- Robot execution is often driven by mathematical optimisation criteria, but:

Conceptual Constraints

- Robot execution is often driven by mathematical optimisation criteria, but:
  - **There is often no explicit guarantee that the execution will actually satisfy those criteria** — there is only a probabilistic guarantee

Conceptual Constraints

- Robot execution is often driven by mathematical optimisation criteria, but:
  - **There is often no explicit guarantee that the execution will actually satisfy those criteria** — there is only a probabilistic guarantee
  - **It can sometimes be difficult to mathematically express criteria in a satisfactory way** — particularly when criteria are expressed by non-expert robot users

Conceptual Constraints

- Robot execution is often driven by mathematical optimisation criteria, but:
  - **There is often no explicit guarantee that the execution will actually satisfy those criteria** — there is only a probabilistic guarantee
  - **It can sometimes be difficult to mathematically express criteria in a satisfactory way** — particularly when criteria are expressed by non-expert robot users
- Conceptual constraints make it possible to **communicate an aspect that we want a robot to comply with in a qualitative manner**
  - Verifying the compliance with a qualitative criterion can often — but not always — be straightforward to implement as well

Conceptual Constraints

- Robot execution is often driven by mathematical optimisation criteria, but:
  - **There is often no explicit guarantee that the execution will actually satisfy those criteria** — there is only a probabilistic guarantee
  - **It can sometimes be difficult to mathematically express criteria in a satisfactory way** — particularly when criteria are expressed by non-expert robot users
- Conceptual constraints make it possible to **communicate an aspect that we want a robot to comply with in a qualitative manner**
  - Verifying the compliance with a qualitative criterion can often — but not always — be straightforward to implement as well

A conceptual constraint is a qualitative criterion that a robot should satisfy during learning or acting

Conceptual Constraints in Robot Verification and Validation

- **Most of the rules that autonomous robots will need to comply with are expressed in a qualitative way** (regulations, laws, best practices)

Conceptual Constraints in Robot Verification and Validation

- **Most of the rules that autonomous robots will need to comply with are expressed in a qualitative way** (regulations, laws, best practices)
- Equipping a robot with an ability to explicitly take conceptual constraints into account during execution can thus:

Conceptual Constraints in Robot Verification and Validation

- **Most of the rules that autonomous robots will need to comply with are expressed in a qualitative way** (regulations, laws, best practices)
- Equipping a robot with an ability to explicitly take conceptual constraints into account during execution can thus:
  - Simplify the process of showing that the robot actually complies with existing rules and regulations (**compliance by design**)

Conceptual Constraints in Robot Verification and Validation

- **Most of the rules that autonomous robots will need to comply with are expressed in a qualitative way** (regulations, laws, best practices)
- Equipping a robot with an ability to explicitly take conceptual constraints into account during execution can thus:
  - Simplify the process of showing that the robot actually complies with existing rules and regulations (**compliance by design**)
  - Make it possible to more easily adapt the robot's behaviour (**changing the qualitative criteria will automatically modify the behaviour**)

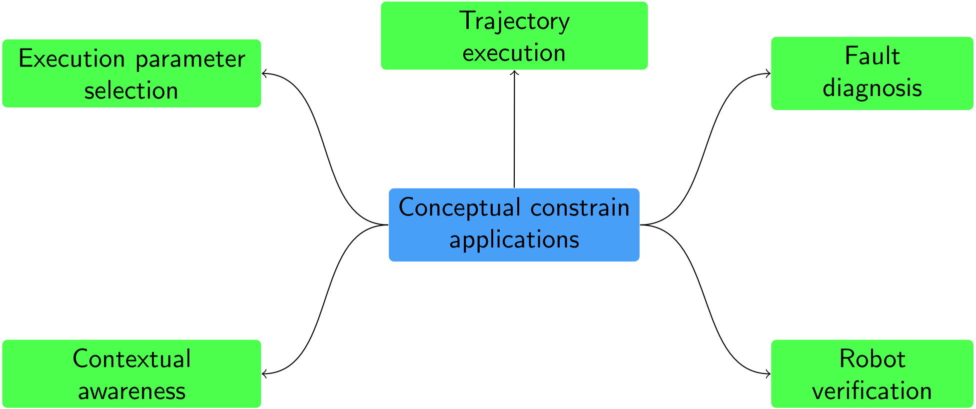
Conceptual Constraints in Robot Verification and Validation

- **Most of the rules that autonomous robots will need to comply with are expressed in a qualitative way** (regulations, laws, best practices)
- Equipping a robot with an ability to explicitly take conceptual constraints into account during execution can thus:
  - Simplify the process of showing that the robot actually complies with existing rules and regulations (**compliance by design**)
  - Make it possible to more easily adapt the robot's behaviour (**changing the qualitative criteria will automatically modify the behaviour**)
- The process of robot verification and validation is thus tightly intertwined with the use of conceptual constraints in a robot's execution process

An Overview of Applications of Conceptual Constraints

Conceptual Constrains Have Many Uses

There is a large number of problems in which qualitative constraints can be used



How to Select Execution Parameters?

- Conceptual constraints can be particularly useful when a robot needs to select parameters for executing its skills

How to Select Execution Parameters?

- Conceptual constraints can be particularly useful when a robot needs to select parameters for executing its skills
- The existence of conceptual constraints can be used to ensure that the robot:

How to Select Execution Parameters?

- Conceptual constraints can be particularly useful when a robot needs to select parameters for executing its skills
- The existence of conceptual constraints can be used to ensure that the robot:
  - **does not violate execution constraints that should be enforced** (e.g. an egg should not be squeezed too much or it will break)

How to Select Execution Parameters?

- Conceptual constraints can be particularly useful when a robot needs to select parameters for executing its skills
- The existence of conceptual constraints can be used to ensure that the robot:
  - **does not violate execution constraints that should be enforced** (e.g. an egg should not be squeezed too much or it will break)
  - **satisfies some explicit desirability criteria about the execution** (e.g. a glass should not be grasped too close to the rim)

How to Select Execution Parameters?

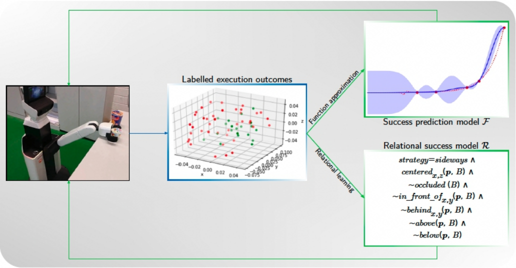
- Conceptual constraints can be particularly useful when a robot needs to select parameters for executing its skills
- The existence of conceptual constraints can be used to ensure that the robot:
  - **does not violate execution constraints that should be enforced** (e.g. an egg should not be squeezed too much or it will break)
  - **satisfies some explicit desirability criteria about the execution** (e.g. a glass should not be grasped too close to the rim)
- A side effect of associating the parameter selection process with conceptual constraints is that **the selection becomes explainable**
  - Explainability is particularly relevant when a robot closely cooperates with human partners

Modelling Execution Success Through Relations

One use of conceptual constraints is to model qualitative criteria that should be satisfied so that the execution of a given skill succeeds

Modelling Execution Success Through Relations

One use of conceptual constraints is to model qualitative criteria that should be satisfied so that the execution of a given skill succeeds



A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "A Hybrid Skill Parameterisation Model Combining Symbolic and Subsymbolic Elements for Introspective Robots," *Robotics and Autonomous Systems*, vol. 161, p. 104350:1–22, Mar. 2023. Available: https://doi.org/10.1016/j.robot.2022.104350

- This is the objective of execution models:

$$\mathcal{M} = (\mathcal{R}, \mathcal{F})$$

- Here, $\mathcal{R}$ is a **conjunction of preconditions under which the execution has been observed to succeed**, while $\mathcal{F}$ models **how likely the execution is to succeed if given parameters are selected for execution**

- The precondition model is a set of constraints on the execution parameters

Parameter Sampling Using an Execution Model

Greedy execution parameter sampling

function $\textsc{sampleParameters}(\mathcal{M}, X, \boldsymbol{e}, \boldsymbol{c}, \epsilon_{\mathcal{M}}, \Sigma_{\tau}, \rho)$
    if $\boldsymbol{c} \neq \varnothing$ **then**
        $X \leftarrow \texttt{knn}(X, \boldsymbol{c})$
        $\boldsymbol{e} = \boldsymbol{e}_X$
    $\hat{\boldsymbol{\tau}} \leftarrow \varnothing$
    $sample\_found \leftarrow$ **false**
    while $sample\_found =$ **false do**
        $\hat{\boldsymbol{x}} \leftarrow \texttt{sample}(\mathcal{F}, X, \boldsymbol{e}, \epsilon_{\mathcal{M}})$
        $\hat{\boldsymbol{\tau}} \leftarrow \mathcal{N}(\hat{\boldsymbol{x}}, \Sigma_{\tau})$
        if $\texttt{verifyPreconditions}(\mathcal{R}, \hat{\boldsymbol{\tau}}, \rho)$ **then**
            $sample\_found \leftarrow$ **true**
    return $\hat{\boldsymbol{\tau}}$

A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "A Hybrid Skill Parameterisation Model Combining Symbolic and Subsymbolic Elements for Introspective Robots," *Robotics and Autonomous Systems*, vol. 161, p. 104350:1–22, Mar. 2023. Available: https://doi.org/10.1016/j.robot.2022.104350

- Given an execution model, parameters can be sampled so that they (i) **maximise the execution success** and (ii) **do not violate the precondition model $\mathcal{R}$**
- The precondition model thus serves the purpose of **filtering out unsuitable execution parameters**

Using Predicate Models for Parameter Sampling

- It is possible to sample **parameters that satisfy particular constraints by using learned predicate models**

Using Predicate Models for Parameter Sampling

- It is possible to sample **parameters that satisfy particular constraints by using learned predicate models**

- The probability that a given relational sentence S is satisfied for a given parameter sample G can be represented by a product over the probabilities of the individual predicates C_i:

$$P(S|G) = \prod_{i=1}^{n} P(C_i|G)$$

Using Predicate Models for Parameter Sampling

- It is possible to sample **parameters that satisfy particular constraints by using learned predicate models**
- The probability that a given relational sentence S is satisfied for a given parameter sample G can be represented by a product over the probabilities of the individual predicates C_i:

$$P(S|G) = \prod_{i=1}^{n} P(C_i|G)$$

- The probability that an individual predicate is satisfied can be learned from training data:

Using Predicate Models for Parameter Sampling

- It is possible to sample **parameters that satisfy particular constraints by using learned predicate models**
- The probability that a given relational sentence S is satisfied for a given parameter sample G can be represented by a product over the probabilities of the individual predicates C_i:

$$P(S|G) = \prod_{i=1}^{n} P(C_i|G)$$

- The probability that an individual predicate is satisfied can be learned from training data:

$$P(C|G_C) = \frac{P(G_C|C)}{P(G_C|C) + P(G_C|\neg C)}$$

Using Predicate Models for Parameter Sampling

- It is possible to sample **parameters that satisfy particular constraints by using learned predicate models**
- The probability that a given relational sentence S is satisfied for a given parameter sample G can be represented by a product over the probabilities of the individual predicates C_i:

$$P(S|G) = \prod_{i=1}^{n} P(C_i|G)$$

- The probability that an individual predicate is satisfied can be learned from training data:

$$P(C|G_C) = \frac{P(G_C|C)}{P(G_C|C) + P(G_C|\neg C)} \qquad P(G_C|C) = \frac{1}{n^+}\sum_{i=1}^{n^+} e^{\frac{-\|G_i^+ - G_C\|^2}{h^2}}$$

Using Predicate Models for Parameter Sampling

- It is possible to sample **parameters that satisfy particular constraints by using learned predicate models**
- The probability that a given relational sentence S is satisfied for a given parameter sample G can be represented by a product over the probabilities of the individual predicates C_i:

$$P(S|G) = \prod_{i=1}^{n} P(C_i|G)$$

- The probability that an individual predicate is satisfied can be learned from training data:

$$P(C|G_C) = \frac{P(G_C|C)}{P(G_C|C) + P(G_C|\neg C)} \qquad P(G_C|C) = \frac{1}{n^+} \sum_{i=1}^{n^+} e^{\frac{-\|G_i^+ - G_C\|^2}{h^2}} \qquad P(G_C|\neg C) = \frac{1}{n^-} \sum_{i=1}^{n^-} e^{-\frac{\|G_i^- - G_C\|^2}{h^2}}$$

Using Predicate Models for Parameter Sampling

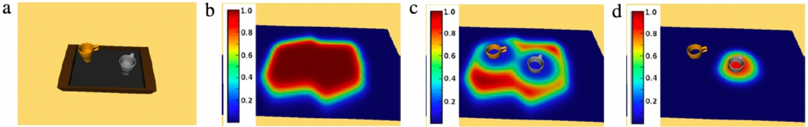
- It is possible to sample **parameters that satisfy particular constraints by using learned predicate models**

- The probability that a given relational sentence S is satisfied for a given parameter sample G can be represented by a product over the probabilities of the individual predicates C_i:

$$P(S|G) = \prod_{i=1}^{n} P(C_i|G)$$

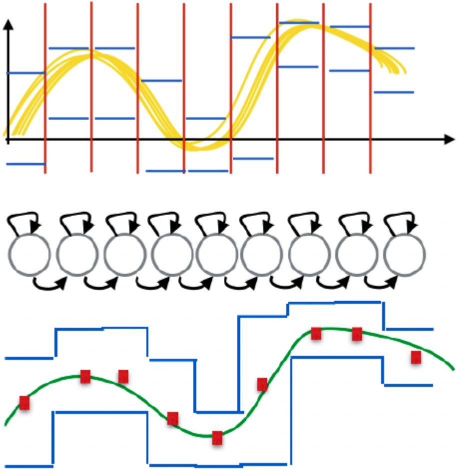
- The probability that an individual predicate is satisfied can be learned from training data:

$$P(C|G_C) = \frac{P(G_C|C)}{P(G_C|C) + P(G_C|\neg C)} \qquad P(G_C|C) = \frac{1}{n^+}\sum_{i=1}^{n^+} e^{\frac{-\|G_i^+ - G_C\|^2}{h^2}} \qquad P(G_C|\neg C) = \frac{1}{n^-}\sum_{i=1}^{n^-} e^{-\frac{\|G_i^- - G_C\|^2}{h^2}}$$



R. Dearden and C. Burbridge, "Manipulation Planning using Learned Symbolic State Abstractions," *Robotics and Autonomous Systems*, vol. 62, no. 3, pp. 355–365, 2014. Available: https://doi.org/10.1016/j.robot.2013.09.015

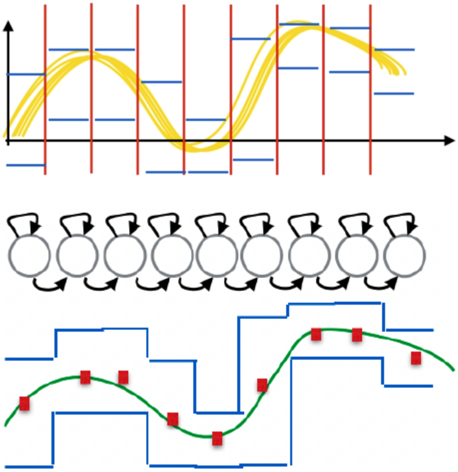
Trajectory Execution with Constraints



A. Haidu, D. Kohlsdorf and M. Beetz, "Learning action failure models from interactive physics-based simulations," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2015, pp. 5370–5375. Available: https://doi.org/10.1109/IROS.2015.7354136

- Conceptual constraints can also be used when **sampling trajectories to be executed — as criteria that the trajectory should not violate**

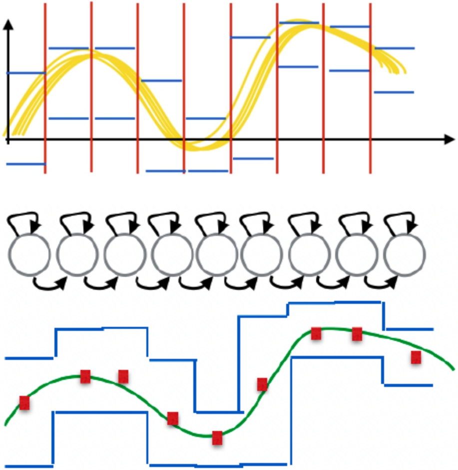
Trajectory Execution with Constraints



A. Haidu, D. Kohlsdorf and M. Beetz, "Learning action failure models from interactive physics-based simulations," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2015, pp. 5370–5375. Available: https://doi.org/10.1109/IROS.2015.7354136

- Conceptual constraints can also be used when **sampling trajectories to be executed — as criteria that the trajectory should not violate**
- In the simplest case, such criteria can be expressed via **envelopes that trajectory samples may not go beyond** — for instance, an envelope can be defined by one standard deviation from the mean trajectory

Trajectory Execution with Constraints



A. Haidu, D. Kohlsdorf and M. Beetz, "Learning action failure models from interactive physics-based simulations," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2015, pp. 5370–5375. Available: https://doi.org/10.1109/IROS.2015.7354136

- Conceptual constraints can also be used when **sampling trajectories to be executed — as criteria that the trajectory should not violate**
- In the simplest case, such criteria can be expressed via **envelopes that trajectory samples may not go beyond** — for instance, an envelope can be defined by one standard deviation from the mean trajectory
- **Constraints can be used to detect failures during trajectory execution** — points that fall outside of the envelope can be used as indications of failures and can trigger a subsequent recovery behaviour

High-Level Trajectory Execution Constraints

- Conceptual constraints for trajectory execution can also be expressed at a "higher level" — in terms of entities involved in the execution (e.g. a plate should not be turned upside down)

High-Level Trajectory Execution Constraints

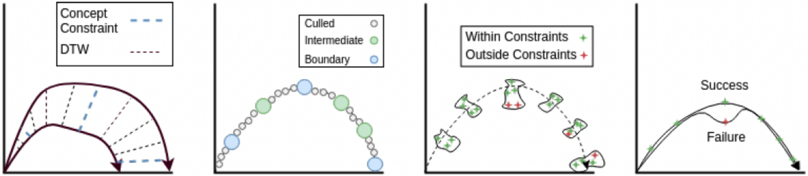
- Conceptual constraints for trajectory execution can also be expressed at a "higher level" — in terms of entities involved in the execution (e.g. a plate should not be turned upside down)
- This is useful in the context of learning from demonstration (as in your previous assignment) and **can be used to inject knowledge that is not explicitly conveyed through the demonstrations** (which might only be focused on the motion to be acquired)

High-Level Trajectory Execution Constraints

- Conceptual constraints for trajectory execution can also be expressed at a "higher level" — in terms of entities involved in the execution (e.g. a plate should not be turned upside down)
- This is useful in the context of learning from demonstration (as in your previous assignment) and **can be used to inject knowledge that is not explicitly conveyed through the demonstrations** (which might only be focused on the motion to be acquired)
- Different criteria may need to be active at different points during the execution —– a trajectory can be executed as a collection of segments, each of which is governed by a different qualitative constraint

High-Level Trajectory Execution Constraints

- Conceptual constraints for trajectory execution can also be expressed at a "higher level" — in terms of entities involved in the execution (e.g. a plate should not be turned upside down)
- This is useful in the context of learning from demonstration (as in your previous assignment) and **can be used to inject knowledge that is not explicitly conveyed through the demonstrations** (which might only be focused on the motion to be acquired)
- Different criteria may need to be active at different points during the execution — a trajectory can be executed as a collection of segments, each of which is governed by a different qualitative constraint



C. Mueller, J. Venicx and B. Hayes, "Robust Robot Learning from Demonstration and Skill Repair Using Conceptual Constraints," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 6029–6036. Available: https://doi.org/10.1109/IROS.2018.8594133

Learning of Planning Models

- Another use of qualitative information is for **learning action models that can be used for task planning** (i.e. for learning the actions' preconditions and/or effects)

Learning of Planning Models

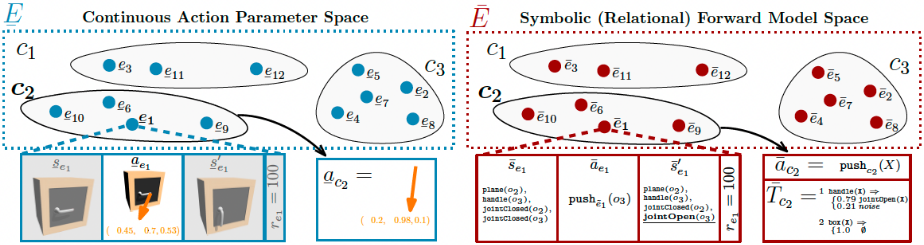
- Another use of qualitative information is for **learning action models that can be used for task planning** (i.e. for learning the actions' preconditions and/or effects)
- This can be achieved by **associating continuous actions with the corresponding relational descriptions of the preconditions and effects**, and learning action models by minimising a loss function on the models' predictive capability

Learning of Planning Models

- Another use of qualitative information is for **learning action models that can be used for task planning** (i.e. for learning the actions' preconditions and/or effects)
- This can be achieved by **associating continuous actions with the corresponding relational descriptions of the preconditions and effects**, and learning action models by minimising a loss function on the models' predictive capability
- Planning using the learned models ensures that the plans are grounded in the robot's own experiences

Learning of Planning Models

- Another use of qualitative information is for **learning action models that can be used for task planning** (i.e. for learning the actions' preconditions and/or effects)
- This can be achieved by **associating continuous actions with the corresponding relational descriptions of the preconditions and effects**, and learning action models by minimising a loss function on the models' predictive capability
- Planning using the learned models ensures that the plans are grounded in the robot's own experiences



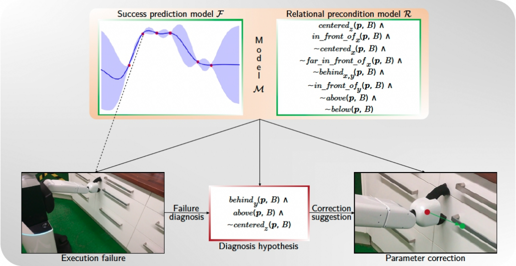
S. Höfer and O. Brock, "Coupled learning of action parameters and forward models for manipulation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016, pp. 3893–3899. Available: https://doi.org/10.1109/IROS.2016.7759573

Fault Diagnosis Using Conceptual Constraints

Fault diagnosis can also benefit from information about conceptual constraints describing a robot's execution

Fault Diagnosis Using Conceptual Constraints

Fault diagnosis can also benefit from information about conceptual constraints describing a robot's execution

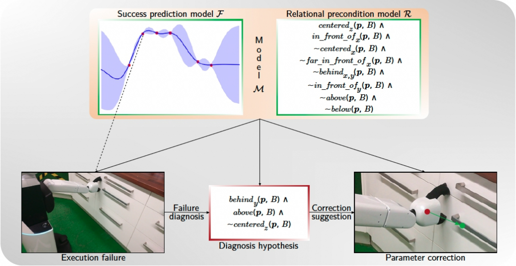


A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "A Hybrid Skill Parameterisation Model Combining Symbolic and Subsymbolic Elements for Introspective Robots," *Robotics and Autonomous Systems*, vol. 161, p. 104350:1–22, Mar. 2023. Available: https://doi.org/10.1016/j.robot.2022.104350

- Failure analysis can particularly be performed by **looking for parameters that lead to a violation of a precondition model**

Fault Diagnosis Using Conceptual Constraints

Fault diagnosis can also benefit from information about conceptual constraints describing a robot's execution



A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "A Hybrid Skill Parameterisation Model Combining Symbolic and Subsymbolic Elements for Introspective Robots," *Robotics and Autonomous Systems*, vol. 161, p. 104350:1–22, Mar. 2023. Available: https://doi.org/10.1016/j.robot.2022.104350

- Failure analysis can particularly be performed by **looking for parameters that lead to a violation of a precondition model**
- Similarly, the search for parameters that correct an execution failure can be informed by conceptual constraints — by **preventing correction candidates that do not satisfy the qualitative preconditions**

Qualitative Context Descriptions

- Particularly in human-centred environments, contextual awareness is an important requirement for a cognitive agent

Qualitative Context Descriptions

- Particularly in human-centred environments, contextual awareness is an important requirement for a cognitive agent
- **Execution context is often best described using qualitative descriptions** (e.g. for object hand-over case, the context can describe the posture of the person as standing, sitting, or lying down)

Qualitative Context Descriptions

- Particularly in human-centred environments, contextual awareness is an important requirement for a cognitive agent
- **Execution context is often best described using qualitative descriptions** (e.g. for object hand-over case, the context can describe the posture of the person as standing, sitting, or lying down)
- **Having a qualitative context description can enable a robot to exhibit context-aware behaviour** — for instance, by selecting skill parameters that are suitable for a given context (e.g. an object should be handed over with care when a person is lying down)

Qualitative Context Descriptions

- Particularly in human-centred environments, contextual awareness is an important requirement for a cognitive agent
- **Execution context is often best described using qualitative descriptions** (e.g. for object hand-over case, the context can describe the posture of the person as standing, sitting, or lying down)
- **Having a qualitative context description can enable a robot to exhibit context-aware behaviour** — for instance, by selecting skill parameters that are suitable for a given context (e.g. an object should be handed over with care when a person is lying down)

A. F. Abdelrahman, A. Mitrevski, and P. G. Plöger, "Context-Aware Task Execution Using Apprenticeship Learning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1329–1335, 2020. Available: https://doi.org/10.1109/ICRA40945.2020.9197476

Property-Based Robot Testing

Property-Based Testing

- Property-based testing is a software testing technique that is concerned with verifying the correctness of a given program[1]

[1] A. Santos, A. Cunha, and N. Macedo, "Property-Based Testing for the Robot Operating System," in *ACM Joint European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2018. Available: https://doi.org/10.1145/3278186.3278195

Property-Based Testing

- Property-based testing is a software testing technique that is concerned with verifying the correctness of a given program[1]
- The overall idea behind the technique is rather simple:

[1] A. Santos, A. Cunha, and N. Macedo, "Property-Based Testing for the Robot Operating System," in *ACM Joint European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2018. Available: https://doi.org/10.1145/3278186.3278195

Property-Based Testing

- Property-based testing is a software testing technique that is concerned with verifying the correctness of a given program[1]
- The overall idea behind the technique is rather simple:
  - **Generate a set X of n data samples from some allowed input space**

[1] A. Santos, A. Cunha, and N. Macedo, "Property-Based Testing for the Robot Operating System," in *ACM Joint European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2018. Available: https://doi.org/10.1145/3278186.3278195

Property-Based Testing

- Property-based testing is a software testing technique that is concerned with verifying the correctness of a given program[1]
- The overall idea behind the technique is rather simple:
  - **Generate a set X of n data samples from some allowed input space**
  - **Run a component under testing for each $x_i, 1 \leq i \leq n$ and obtain an output o_i**

[1] A. Santos, A. Cunha, and N. Macedo, "Property-Based Testing for the Robot Operating System," in *ACM Joint European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2018. Available: https://doi.org/10.1145/3278186.3278195

Property-Based Testing

- Property-based testing is a software testing technique that is concerned with verifying the correctness of a given program[1]
- The overall idea behind the technique is rather simple:
  - **Generate a set X of n data samples from some allowed input space**
  - **Run a component under testing for each $x_i, 1 \leq i \leq n$ and obtain an output o_i**
  - **For each trial, verify that o_i satisfies a set of desired properties P**

[1] A. Santos, A. Cunha, and N. Macedo, "Property-Based Testing for the Robot Operating System," in *ACM Joint European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2018. Available: https://doi.org/10.1145/3278186.3278195

Property-Based Testing

- Property-based testing is a software testing technique that is concerned with verifying the correctness of a given program[1]
- The overall idea behind the technique is rather simple:
  - **Generate a set X of n data samples from some allowed input space**
  - **Run a component under testing for each $x_i, 1 \leq i \leq n$ and obtain an output o_i**
  - **For each trial, verify that o_i satisfies a set of desired properties P**
- Property-based testing is a generalisation of unit testing, but is built with functional components in mind

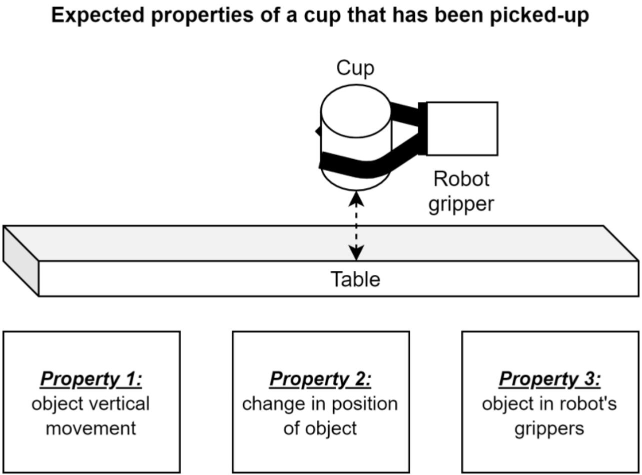
[1] A. Santos, A. Cunha, and N. Macedo, "Property-Based Testing for the Robot Operating System," in *ACM Joint European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2018. Available: https://doi.org/10.1145/3278186.3278195

Property-Based Testing

- Property-based testing is a software testing technique that is concerned with verifying the correctness of a given program[1]
- The overall idea behind the technique is rather simple:
  - **Generate a set X of n data samples from some allowed input space**
  - **Run a component under testing for each $x_i, 1 \leq i \leq n$ and obtain an output o_i**
  - **For each trial, verify that o_i satisfies a set of desired properties P**
- Property-based testing is a generalisation of unit testing, but is built with functional components in mind
- **Properties can be observed as conceptual constraints** that need to be satisfied and which are verified during the testing process

[1] A. Santos, A. Cunha, and N. Macedo, "Property-Based Testing for the Robot Operating System," in *ACM Joint European Software Eng. Conf. and Symp. on the Foundations of Software Eng. (ESEC/FSE)*, 2018. Available: https://doi.org/10.1145/3278186.3278195

Properties Example: Object Grasping



S. O. Sohail, A. Mitrevski, N. Hochgeschwender and P. G. Plöger, "Property-Based Testing in Simulation for Verifying Robot Action Execution in Tabletop Manipulation," in *Proc. European Conf. Mobile Robots (ECMR)*, 2021, pp. 1–7. Available: https://doi.org/10.1109/ECMR50962.2021.9568837

Property-Based Testing Example

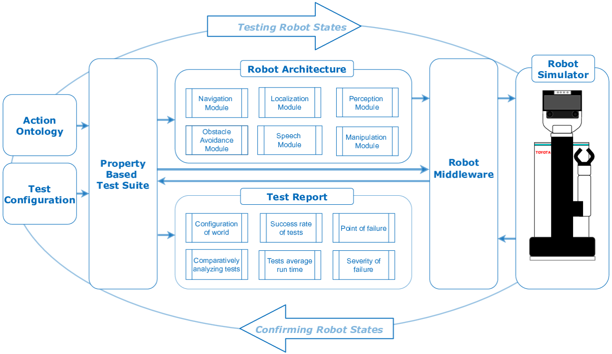
- The example[2] on the right illustrates a property-based testing program for a pick-and-place robot scenario
- Note: Programming languages have different libraries for property-based testing, but most of these are derived from the QuickCheck[3] library in the Haskell programming language

```
from hypothesis import given, strategies as st
@given(generate_placing_problem(generate_surface(),
                                st.integers(1, 10)),
       generate_grasping_problem(generate_surface(),
                                 st.integers(1, 10),
                                 choose_grasping_object()),
       position_robot())
def test_pick_and_place(placing_surface, objects_on_placing_surface,
                        grasping_surface, objects_on_grasping_surface,
                        object_to_grasp, robot_pose):
    localise_robot(robot_pose)

    grasp_result = grasp(object_to_grasp)
    assert object_in_gripper(grasp_result)
    for x in objects_on_grasping_surface:
        assert object_on_surface(x, grasping_surface)

    move_to(placing_surface)
    assert robot_at(placing_surface)

    place_result = place(object_to_grasp)
    assert object_on_surface(object_to_grasp, placing_surface)
    for x in objects_on_placing_surface:
        assert object_on_surface(x, placing_surface)
```

[2] Example using Hypothesis: https://hypothesis.readthedocs.io/en/latest/index.html

[3] QuickCheck: https://hackage.haskell.org/package/QuickCheck

Property-Based Testing in Simulation



S. O. Sohail, A. Mitrevski, N. Hochgeschwender and P. G. Plöger, "Property-Based Testing in Simulation for Verifying Robot Action Execution in Tabletop Manipulation," in *Proc. European Conf. Mobile Robots (ECMR)*, 2021, pp. 1–7. Available: https://doi.org/10.1109/ECMR50962.2021.9568837

- Property-based testing can be used to **facilitate simulation-based testing of robots** — for instance, to verify the successful execution of robot actions
- **The informativeness of testing depends on the ability to generate representative and exhaustive test scenarios**

Summary: Conceptual Constraints

- Conceptual constraints are criteria that are expressed qualitatively and which convey information about certain aspects of a robot's execution process
- There are a variety of uses of conceptual constraints, such as for execution parameter selection, trajectory execution, context-aware acting, as well as fault detection and diagnosis
- Conceptual constraints can also be used for robot testing, concretely in the case of property-based testing, in order to verify that a robot satisfies particular execution requirements