Towards a Unifying Framework for Uncertainty in CPS

for Jan Peleska

Jim Woodcock

University of York | Aarhus University

3rd March 2023

Outline

Introduction

Motivation

Prism’s Semantics

Probabilistic Predicative Programming

Conclusions

Jan Peleska

- This talk is dedicated with affection to Jan Peleska on his 65th birthday.
- We discuss a unifying theory of uncertainty in robotics and CPS.
- We use Hoare & He's UTP and Hehner's probabilistic predicative programming.
- This is a long-term research agenda at York and Aarhus universities.
- We start with a semantics for Prism and end with many questions.

Jan Peleska

- This talk is dedicated with affection to Jan Peleska on his 65th birthday.
- We discuss a unifying theory of uncertainty in robotics and CPS.
- We use Hoare & He's UTP and Hehner's probabilistic predicative programming.
- This is a long-term research agenda at York and Aarhus universities.
- We start with a semantics for Prism and end with many questions.

Jan Peleska

- This talk is dedicated with affection to Jan Peleska on his 65th birthday.
- We discuss a unifying theory of uncertainty in robotics and CPS.
- We use Hoare & He's UTP and Hehner's probabilistic predicative programming.
- This is a long-term research agenda at York and Aarhus universities.
- We start with a semantics for Prism and end with many questions.

Jan Peleska

- This talk is dedicated with affection to Jan Peleska on his 65th birthday.
- We discuss a unifying theory of uncertainty in robotics and CPS.
- We use Hoare & He's UTP and Hehner's probabilistic predicative programming.
- This is a long-term research agenda at York and Aarhus universities.
- We start with a semantics for Prism and end with many questions.

Jan Peleska

- This talk is dedicated with affection to Jan Peleska on his 65th birthday.
- We discuss a unifying theory of uncertainty in robotics and CPS.
- We use Hoare & He’s UTP and Hehner’s probabilistic predicative programming.
- This is a long-term research agenda at York and Aarhus universities.
- We start with a semantics for Prism and end with many questions.

Jan Peleska

- This talk is dedicated with affection to Jan Peleska on his 65th birthday.
- We discuss a unifying theory of uncertainty in robotics and CPS.
- We use Hoare & He's UTP and Hehner's probabilistic predicative programming.
- This is a long-term research agenda at York and Aarhus universities.
- We start with a semantics for Prism and end with many questions.

Jan Peleska

- This talk is dedicated with affection to Jan Peleska on his 65th birthday.
- We discuss a unifying theory of uncertainty in robotics and CPS.
- We use Hoare & He's UTP and Hehner's probabilistic predicative programming.
- This is a long-term research agenda at York and Aarhus universities.
- We start with a semantics for Prism and end with many questions.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan's Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan's Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan's Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan's Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan's Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan’s Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

This Paper is Inspired by Jan's Work

- We first met at 09:00 on the 10th of June 1991 at DST in Hamburg.
- DST were developing the Airbus Interphone System.
- Jan made me adapt my course examples and practicals.
- We found a significant error in the Interphone specification.
- Cabin crew could get locked out of a conference call during an emergency.
- We later worked together on European projects, such as INTO-CPS.
- Jan and Wen-ling developed a runtime verification technique for CSP.
- This checks that a system under test satisfies properties over CSP failures.
- This depends on the soundness of the two-way translation.
- With Ana, the four of us formalised the soundness argument.
- Normalised graphs in CSP model checking, action systems, Kripke structures.
- This paper extends that work to probabilistic Kripke structures.
- Objective Unify formalisms and tools for treating uncertainty in robotics.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object’s centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object’s centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

Uncertainty in Robotics

- Autonomous vehicle tries to pass quickly through intersection without signals.
- Counterintuitively, the vehicle slows down instead of accelerating.
- It gathers information on the intentions of pedestrians and other vehicles.
- This information helps the vehicle coordinate its actions with others.
- It achieves its overall goal faster in the long term.

- Robot manipulator tries to push an irregular object to a designated pose.
- The robot must minimise the number of actions.
- It decides not to push the object directly towards the final pose.
- It uses the first pushes to gather information on the object's centre of mass.
- The later pushes are now much more effective.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, ...
- What would a unifying theory for uncertainty look like?

- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.

- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, ...
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.
- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?

- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.

- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.

- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.
- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.
- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.

- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.

- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.
- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?

- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.

- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.

- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.
- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.
- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?

- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.

- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.

- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?
- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.
- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.
- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

A Unifying Framework for Uncertainty?

- pGCL, MDPs, POMDPs, dynamic epistemic logic, epistemic mu-calculus, . . .
- What would a unifying theory for uncertainty look like?

- Research Hypothesis

  We can unify different theories of uncertainty using:

  UTP probabilistic relations. Bayesian semantics. Information theory.

- We focus on a specific domain initially: robot planning.

  Modelling and solving robot decision and control tasks under uncertainty.

  Noisy sensing, imperfect control, environment changes, inaccurate models.

  Localisation and navigation, search and tracking, autonomous driving.

  Multi-robot systems, object manipulation, human-robot interaction.

- Robot reasons about outcomes of actions with limited sensor information.
- Actions have short-term rewards and inform long-term success.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.
- Unifying semantics Powerful enough for SotA modelling languages.
- Denotational semantics Gold standard.
- Operational semantics Soundness wrto denotational semantics.
- Algebraic semantics Derived from opsem soundness proof.
- Programming logic Probabilistic Hoare logic (cf. Hartog & de Vink).
- Refinement theory Refinement calculus (cf. McIver & Morgan).
- Testing theory Testing practical systems (cf. Gaudel TcbFt).
- Mechanisation Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.
- Unifying semantics   Powerful enough for SotA modelling languages.
- Denotational semantics   Gold standard.
- Operational semantics   Soundness wrto denotational semantics.
- Algebraic semantics   Derived from opsem soundness proof.
- Programming logic   Probabilistic Hoare logic (cf. Hartog & de Vink).
- Refinement theory   Refinement calculus (cf. McIver & Morgan).
- Testing theory   Testing practical systems (cf. Gaudel TcbFt).
- Mechanisation   Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.

Unifying semantics	Powerful enough for SotA modelling languages.
Denotational semantics	Gold standard.
Operational semantics	Soundness wrto denotational semantics.
Algebraic semantics	Derived from opsem soundness proof.
Programming logic	Probabilistic Hoare logic (cf. Hartog & de Vink).
Refinement theory	Refinement calculus (cf. McIver & Morgan).
Testing theory	Testing practical systems (cf. Gaudel TcbFt).
Mechanisation	Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.

► Unifying semantics	Powerful enough for SotA modelling languages.
► Denotational semantics	Gold standard.
► Operational semantics	Soundness wrto denotational semantics.
► Algebraic semantics	Derived from opsem soundness proof.
► Programming logic	Probabilistic Hoare logic (cf. Hartog & de Vink).
► Refinement theory	Refinement calculus (cf. McIver & Morgan).
► Testing theory	Testing practical systems (cf. Gaudel TcbFt).
► Mechanisation	Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.
- Unifying semantics Powerful enough for SotA modelling languages.
- Denotational semantics **Gold standard.**
- Operational semantics Soundness wrto denotational semantics.
- Algebraic semantics Derived from opsem soundness proof.
- Programming logic Probabilistic Hoare logic (cf. Hartog & de Vink).
- Refinement theory Refinement calculus (cf. McIver & Morgan).
- Testing theory Testing practical systems (cf. Gaudel TcbFt).
- Mechanisation Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.
- Unifying semantics Powerful enough for SotA modelling languages.
- Denotational semantics **Gold standard.**
- Operational semantics Soundness wrto denotational semantics.
- Algebraic semantics Derived from opsem soundness proof.
- Programming logic Probabilistic Hoare logic (cf. Hartog & de Vink).
- Refinement theory Refinement calculus (cf. McIver & Morgan).
- Testing theory Testing practical systems (cf. Gaudel TcbFt).
- Mechanisation Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.
- Unifying semantics Powerful enough for SotA modelling languages.
- Denotational semantics **Gold standard.**
- Operational semantics Soundness wrto denotational semantics.
- Algebraic semantics Derived from opsem soundness proof.
- Programming logic Probabilistic Hoare logic (cf. Hartog & de Vink).
- Refinement theory Refinement calculus (cf. McIver & Morgan).
- Testing theory Testing practical systems (cf. Gaudel TcbFt).
- Mechanisation Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.

Unifying semantics	Powerful enough for SotA modelling languages.
Denotational semantics	**Gold standard.**
Operational semantics	Soundness wrto denotational semantics.
Algebraic semantics	Derived from opsem soundness proof.
Programming logic	Probabilistic Hoare logic (cf. Hartog & de Vink).
Refinement theory	Refinement calculus (cf. McIver & Morgan).
Testing theory	Testing practical systems (cf. Gaudel TcbFt).
Mechanisation	Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.
- Unifying semantics Powerful enough for SotA modelling languages.
- Denotational semantics **Gold standard.**
- Operational semantics Soundness wrto denotational semantics.
- Algebraic semantics Derived from opsem soundness proof.
- Programming logic Probabilistic Hoare logic (cf. Hartog & de Vink).
- Refinement theory Refinement calculus (cf. McIver & Morgan).
- Testing theory Testing practical systems (cf. Gaudel TcbFt).
- Mechanisation Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.
- Unifying semantics Powerful enough for SotA modelling languages.
- Denotational semantics **Gold standard.**
- Operational semantics Soundness wrto denotational semantics.
- Algebraic semantics Derived from opsem soundness proof.
- Programming logic Probabilistic Hoare logic (cf. Hartog & de Vink).
- Refinement theory Refinement calculus (cf. McIver & Morgan).
- Testing theory Testing practical systems (cf. Gaudel TcbFt).
- Mechanisation Implementation in Isabelle/UTP.

Unifying Semantics for Prism

- Towards a specification-oriented semantics for proof and refinement.
- Start with DTMCs. Extend to MDPs, POMDPs, CTMCs, PAs, PTAs, POPTAs.

Unifying semantics	Powerful enough for SotA modelling languages.
Denotational semantics	**Gold standard.**
Operational semantics	Soundness wrto denotational semantics.
Algebraic semantics	Derived from opsem soundness proof.
Programming logic	Probabilistic Hoare logic (cf. Hartog & de Vink).
Refinement theory	Refinement calculus (cf. McIver & Morgan).
Testing theory	Testing practical systems (cf. Gaudel TcbFt).
Mechanisation	Implementation in Isabelle/UTP.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

This Talk

- Small motivating example of Prism DTMC.
- Discussion on why we need a formal semantics for Prism.
- Nonprobabilistic semantics: Unity and Kripke semantics.
- Existing system module semantics for Prism.
- Predicative Programming technique as a Kripke semantics.
- Probabilistic Predicative Programming technique as a probabilistic semantics.
- Probabilistic specifications and killer robot example.
- Where we go from here.

Outline

Introduction

Motivation

Prism’s Semantics

Probabilistic Predicative Programming

Conclusions

Prism DTMC Example

- Throw a pair of six-sided dice until they are equal. How long will this take?

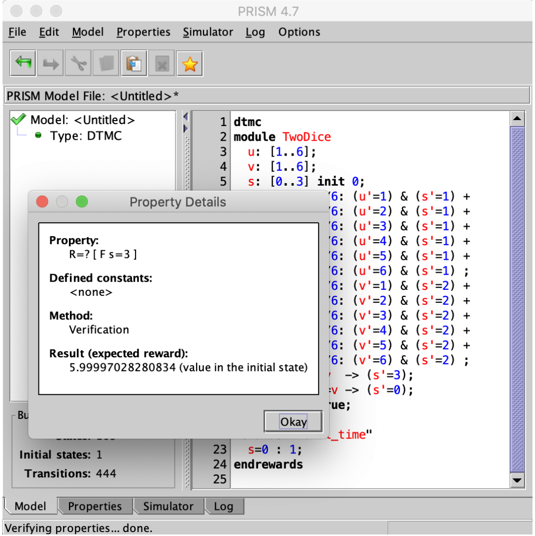
```
dtmc

module TwoDice
  u: [1..6];
  v: [1..6];
  s: [0..3] init 0;
  [] s=0 -> 1/6: (u'=1) & (s'=1) + 1/6: (u'=2) & (s'=1) + 1/6: (u'=3) & (s'=1) +
            1/6: (u'=4) & (s'=1) + 1/6: (u'=5) & (s'=1) + 1/6: (u'=6) & (s'=1) ;
  [] s=1 -> 1/6: (v'=1) & (s'=2) + 1/6: (v'=2) & (s'=2) + 1/6: (v'=3) & (s'=2) +
            1/6: (v'=4) & (s'=2) + 1/6: (v'=5) & (s'=2) + 1/6: (v'=6) & (s'=2) ;
  [] s=2 & u=v  -> (s'=3);
  [] s=2 & u!=v -> (s'=0);
  [] s=3 -> true;
endmodule

rewards "total_time"
  s=0 : 1;
endrewards
```

Prism Check

- We have a Prism model.
- But what properties does it have?
- How many throws of the dice-pair?
- How many throws, on average, do we need to terminate?
- Reward structure gives time steps.
- What's the expected time taken to reach, from the initial state, `s=3`?
- Prism says: you need `5.99997028280834` throws.
- But what if we have 10 dice?
- How many throws do we now need?



PRISM Model File: <Untitled>*

Model: <Untitled>
Type: DTMC

```
dtmc
module TwoDice
  u: [1..6];
  v: [1..6];
  s: [0..3] init 0;
                 6: (u'=1) & (s'=1) +
                 6: (u'=2) & (s'=1) +
                 6: (u'=3) & (s'=1) +
                 6: (u'=4) & (s'=1) +
                 6: (u'=5) & (s'=1) +
                 6: (u'=6) & (s'=1) ;
                 6: (v'=1) & (s'=2) +
                 6: (v'=2) & (s'=2) +
                 6: (v'=3) & (s'=2) +
                 6: (v'=4) & (s'=2) +
                 6: (v'=5) & (s'=2) +
                 6: (v'=6) & (s'=2) ;
                   -> (s'=3);
                 v -> (s'=0);
                 ue;

                 _time"
  s=0 : 1;
endrewards
```

Property Details

Property:
R=? [F s=3]

Defined constants:
<none>

Method:
Verification

Result (expected reward):
5.99997028280834 (value in the initial state)

Bu

Initial states: 1

Transitions: 444

Verifying properties... done.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Why Do We Need another Formal Semantics for Prism?

Prism's process algebra operators

- CSP-based. But some aspects are only syntactic, not semantic.
- Prism action labels are not CSP Events.
- Prism deadlock is not CSP Deadlock.
- Prism hiding is not CSP Hiding.

More powerful verification and validation of probabilistic systems

- Refinement Theory Correctness by construction.
- Assertions Theorem proving. Design by contract. Runtime checking.
- Tool integration Model checking + theorem proving.
- Testing Theory (Probabilistic) testing can be formal, too.

Outline

Introduction

Motivation

Prism’s Semantics

Probabilistic Predicative Programming

Conclusions

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Kripke Structure

- Describes models with propositionally labelled states.
- Temporal logic semantics traditionally given using Kripke structures.
- Structure consists principally of a transition relation.
- Nodes represent reachable system states. Edges represent state transitions.
- Labelling function maps a node to a set of properties holding in that state.
- Why use Kripke structures?

  They represent closed finite-state models with independent state encoding.
- This captures the notion of observability to relate to actual executions.
- An observer might not be able to read all state variables.
- Trace: sequence of observable parts of states.

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u.`
- Sub-state space $S_c = \{ s \in S \mid s \models g \}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \to S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z)`.

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[] g->u`.
- Sub-state space $S_c = \{ s \in S \mid s \models g \}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \to S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z)`.

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u.`
- Sub-state space $S_c = \{ s \in S \mid s \models g \}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \rightarrow S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z).`

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u.`
- Sub-state space $S_c = \{ s \in S \mid s \models g \}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \to S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z).`

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u.`
- Sub-state space $S_c = \{ s \in S \mid s \models g \}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \rightarrow S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z)`.

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u.`
- Sub-state space $S_c = \{\, s \in S \mid s \models g \,\}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \rightarrow S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z).`

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u`.
- Sub-state space $S_c = \{\, s \in S \mid s \models g \,\}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \to S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z)`.

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u.`
- Sub-state space $S_c = \{ s \in S \mid s \models g \}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \to S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z).`

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u`.
- Sub-state space $S_c = \{ s \in S \mid s \models g \}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \rightarrow S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z)`.

Unity Single Command Semantics

- Prism $\setminus$ Probability $=$ Unity [Chandy & Misra]
- Guarded command $c \in C$ `[]g->u`.
- Sub-state space $S_c = \{\, s \in S \mid s \models g \,\}$ UTP: this is simply g.
- Guard g predicate over variables in V.
- Command u assignments to variables in V.
- Update u of c $u : S_c \rightarrow S$.
- Example `[] x>y -> (x'=y) & (y'=x);`
- Semantics $x > y \wedge (x' = y) \wedge (y' = x) \wedge (z' = z)$.
- Extension variables (unmentioned) stay the same: Example `(z'=z)`.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ g \to p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{ s \in S \mid s \models g \}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \to S$.
- Let $u_j = \bigwedge i : 1 \,..\, m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \,..\, m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \to \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \ \widehat{=}\ \sum j : 1 \,..\, n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a] \;\; g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \,..\, m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \,..\, m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \;\; \widehat{=} \;\; \sum j : 1 \,..\, n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ g \to p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{ s \in S \mid s \models g \}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \to S$.
- Let $u_j = \bigwedge i : 1 \mathinner{\ldotp\ldotp} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{\ldotp\ldotp} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \to \mathbb{R}_{\geq 0}$, for $t \in S$:

  $$\mu_{c,s}(t) \ \widehat{=}\ \sum j : 1 \mathinner{\ldotp\ldotp} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ g \to p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{ s \in S \mid s \models g \}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \to S$.
- Let $u_j = \bigwedge i : 1 \,..\, m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \,..\, m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \to \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \ \widehat{=}\ \sum j : 1 \,..\, n \bullet [\, u_j(s) = t\,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\;\; g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \,..\, m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \,..\, m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

  $$\mu_{c,s}(t) \;\widehat{=}\; \sum j : 1 \,..\, n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \ \widehat{=}\ \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ \ g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

  $$\mu_{c,s}(t) \ \ \widehat{=}\ \ \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$
- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a] \;\; g \to p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \to S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \to \mathbb{R}_{\geq 0}$, for $t \in S$:

  $$\mu_{c,s}(t) \;\widehat{=}\; \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{ s \in S \mid s \models g \}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \ \widehat{=}\ \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ \ g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \ \ \widehat{=}\ \ \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \ \widehat{=}\ \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\;\; g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

$$\mu_{c,s}(t) \;\; \widehat{=} \;\; \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism System Module Semantics

- Single Command Semantics. Assumptions: the same as Unity.
- Command c of C is schematically: $[a]\ \ g \rightarrow p_1 : u_1 + \cdots + p_n : u_n$
- Action label a is needed only for flattening process-algebraic operators.
- Guard g is a predicate over the variables in V.
- g defines global state subset $S_c = \{\, s \in S \mid s \models g \,\}$.
- Update u_j of c: transition assigning values to variables $u_j : S_c \rightarrow S$.
- Let $u_j = \bigwedge i : 1 \mathinner{..} m \bullet (v_i' = e_i)$.
- Each $s \in S_c$, is an m-tuple: $i \in 1 \mathinner{..} m \Rightarrow (ti = e_i(s))$.
- Update u_j in c occurs with probability p_j.
- c defines, for $s \in S_c$, a function $\mu_{c,s} : S \rightarrow \mathbb{R}_{\geq 0}$, for $t \in S$:

  $$\mu_{c,s}(t) \ \ \widehat{=}\ \ \sum j : 1 \mathinner{..} n \bullet [\, u_j(s) = t \,] * p_j$$

- DTMC and MDP syntax guarantees $\mu_{c,s}$ is a probability distribution over S.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

$$\overline{P}(s,t) \;\widehat{=}\; \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

$$P(s,t) \;\widehat{=}\; \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

  $$\overline{P}(s, t) \;\widehat{=}\; \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

  $$P(s, t) \;\widehat{=}\; \overline{P}(s, t) / \sum u : S \bullet \overline{P}(s, u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

$$\overline{P}(s, t) \;\widehat{=}\; \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

$$P(s, t) \;\widehat{=}\; \overline{P}(s, t) / \sum u : S \bullet \overline{P}(s, u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

$$\overline{P}(s,t) \ \widehat{=}\ \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

$$P(s,t) \ \widehat{=}\ \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

  $$\overline{P}(s,t) \ \widehat{=}\ \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

  $$P(s,t) \ \widehat{=}\ \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

  $$\overline{P}(s,t) \ \widehat{=} \ \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

  $$P(s,t) \ \widehat{=} \ \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

$$\overline{P}(s,t) \;\widehat{=}\; \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

$$P(s,t) \;\widehat{=}\; \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

  $$\overline{P}(s,t) \;\widehat{=}\; \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

  $$P(s,t) \;\widehat{=}\; \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

  $$\overline{P}(s,t) \;\widehat{=}\; \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

  $$P(s,t) \;\widehat{=}\; \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

  $$\overline{P}(s,t) \ \widehat{=} \ \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

  $$P(s,t) \ \widehat{=} \ \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Prism DTMC Semantics

- A discrete-time Markov chain is defined by a transition probability matrix.
- First, define the matrix, for any $s, t \in S$:

  $$\overline{P}(s,t) \;\widehat{=}\; \sum c : C \bullet \mu_{c,s}(t)$$

- The rows of $\overline{P}$ may sum to more than 1. Why?
- Local nondeterminism in a module: overlapping guards.
- Prism displays a warning when local nondeterminism is detected in a DTMC.
- Nondeterministic choice is randomised.
- A probability distribution is obtained by normalising $\overline{P}$:

  $$P(s,t) \;\widehat{=}\; \overline{P}(s,t) / \sum u : S \bullet \overline{P}(s,u)$$

- Replaces nondeterminism by uniform probabilistic choice between transitions.

Outline

Introduction

Motivation

Prism's Semantics

Probabilistic Predicative Programming

Conclusions

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[0 \le e \le 1]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e = 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall n, m : 1 .. \infty \bullet 0 \le (1/2)^{n+m} \le 1$
  2. Sum: $(\sum n, m : 1 .. \infty \bullet (1/2)^{n+m}) = 1$

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0 .. \infty \bullet (1/2)^{n+m}) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $(0 \le e \le 1)$
  2. Its sum (for all assignments to v) is 1: $(\sum v \bullet e) = 1$

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall n, m : 1 \,..\, \infty \bullet 0 \le (1/2)^{n+m} \le 1$
  2. Sum: $(\sum n, m : 1 \,..\, \infty \bullet (1/2)^{n+m}) = 1$

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0 \,..\, \infty \bullet (1/2)^{n+m}) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[\,0 \leq e \leq 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e = 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall n, m : 1 \,..\, \infty \bullet 0 \leq (1/2)^{n+m} \leq 1$.
  2. Sum: $(\sum n, m : 1 \,..\, \infty \bullet (1/2)^{n+m}) = 1$.

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0 \,..\, \infty \bullet (1/2)^{n+m}) \neq 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:
  1. Its value (for all assignments to v) is a probability: $[\,0 \le e \le 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e \;=\; 1$.
- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:
  1. Values: $\forall n, m : 1\,..\,\infty \bullet 0 \le (1/2)^{n+m} \le 1$
  2. Sum: $(\sum n, m : 1\,..\,\infty \bullet (1/2)^{n+m}) = 1$
- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0\,..\,\infty \bullet (1/2)^{n+m}) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[\,0 \le e \le 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e = 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall n, m : 1 \mathinner{\ldotp\ldotp} \infty \bullet 0 \le (1/2)^{n+m} \le 1$.
  2. Sum: $(\sum n, m : 1 \mathinner{\ldotp\ldotp} \infty \bullet (1/2)^{n+m}) = 1$.

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0 \mathinner{\ldotp\ldotp} \infty \bullet (1/2)^{n+m}) \ne 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[\,0 \le e \le 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e = 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall n, m : 1 \ldots \infty \bullet 0 \le (1/2)^{n+m} \le 1$
  2. Sum: $(\sum n, m : 1 \ldots \infty \bullet (1/2)^{n+m}) = 1$

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0 \ldots \infty \bullet (1/2)^{n+m}) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[\,0 \le e \le 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e \;=\; 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall\, n, m : 1 \,..\, \infty \bullet 0 \le (1/2)^{n+m} \le 1$.
  2. Sum: $(\sum n, m : 1 \,..\, \infty \bullet (1/2)^{n+m}) \;=\; 1$.

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0 \,..\, \infty \bullet (1/2)^{n+m}) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[\,0 \le e \le 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e \;=\; 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall\, n, m : 1\,..\,\infty \bullet 0 \le (1/2)^{n+m} \le 1$.
  2. Sum: $(\sum n, m : 1\,..\,\infty \bullet (1/2)^{n+m}) \;=\; 1$.

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0\,..\,\infty \bullet (1/2)^{n+m}) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:
  1. Its value (for all assignments to v) is a probability: $[\, 0 \le e \le 1 \,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e \;=\; 1$.
- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:
  1. Values: $\forall\, n, m : 1 \,..\, \infty \bullet 0 \le (1/2)^{n+m} \le 1$.
  2. Sum: $(\sum n, m : 1 \,..\, \infty \bullet (1/2)^{n+m}) \;=\; 1$.
- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\textstyle\sum n, m : 0 \,..\, \infty \bullet (1/2)^{n+m}) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[\,0 \le e \le 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e \;=\; 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall\, n, m : 1\,..\,\infty \bullet 0 \le (1/2)^{n+m} \le 1$.
  2. Sum: $(\sum n, m : 1\,..\,\infty \bullet (1/2)^{n+m}\,) \;=\; 1$.

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0\,..\,\infty \bullet (1/2)^{n+m}\,) = 1$$

Discrete Distribution

- Suppose that e is an expression with free variables v.
- Expression e is a discrete distribution if it satisfies two criteria:

  1. Its value (for all assignments to v) is a probability: $[\,0 \le e \le 1\,]$.
  2. Its sum (for all assignments to v) is 1: $\sum v \bullet e \;=\; 1$.

- Suppose n and m are strictly positive integers.
- Then $(1/2)^{n+m}$ is a distribution because it satisfies the two criteria:

  1. Values: $\forall\, n, m : 1\,..\,\infty \bullet 0 \le (1/2)^{n+m} \le 1$.
  2. Sum: $(\sum n, m : 1\,..\,\infty \bullet (1/2)^{n+m}\,) \;=\; 1$.

- Suppose n and m are nonnegative integers (in contrast to the last example).
- $(1/2)^{n+m}$ is not a distribution, because it fails the second criterion:

$$(\sum n, m : 0\,..\,\infty \bullet (1/2)^{n+m}\,) = 1$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs ¹⁄₁₆ of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$

- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 $\frac{1}{8}$ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs $\frac{1}{16}$ of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1/16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1⁄16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$

- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1⁄16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1⁄16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \ = \ (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \ = \ (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \ = \ 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \ = \ 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs ¹⁄₁₆ of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then
  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$
  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is
  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is
  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1⁄16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1⁄16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1⁄16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \ = \ (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \ = \ (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \ = \ 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \ = \ 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \land (m = 1)$ occurs 1⁄16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$

- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs 1/16 of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 $\frac{1}{8}$ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs $\frac{1}{16}$ of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then

  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$

  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is

  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is

  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Probability Distributions

- Distribution: frequency of occurrence of values of variables.
- Example 2^{-n}: says n has value 3 ⅛ of the time.
- Example $(1/2)^{n+m}$: says state $(n = 3) \wedge (m = 1)$ occurs ¹⁄₁₆ of the time.
- If $n, m : \mathbb{N}_1$ are distributed as $(1/2)^{n+m}$, then
  $$\sum m : \mathbb{N}_1 \bullet (1/2)^{n+m} \;=\; (1/2)^n$$
  gives the frequency of occurrence of values of n.
- Independent variables: product of distributions partitioning variables.
- Example: $(1/2)^{n+m} \;=\; (1/2)^n * (1/2)^m$, so n and m are independent.
- Average value of e as v varies according to distribution p is $\sum v \bullet e * p$.
- Example: average value of n^2 as n varies over $\mathbb{N}_1$ with $(1/2)^n$ is
  $$\sum n : \mathbb{N}_1 \bullet n^2 * (1/2)^n \;=\; 6.$$
- Average value of $n - m$ as n and m vary over $\mathbb{N}_1$ with distribution $(1/2)^{n+m}$ is
  $$\sum n, m : \mathbb{N}_1 \bullet (n - m) * (1/2)^{n+m} \;=\; 0.$$

Normalisation

Definition (Normalisation)

If E's variables are n and m, then

$$\mathbf{N}\,(E) \;\widehat{=}\; E \,/\, \left(\sum n, m \bullet E\right)$$

- Let E be an expression:
  - Whose value (for all assignments of values) is nonnegative.
  - Whose sum (over all assignments of values) is strictly between 0 and ∞.
- Then, the normalisation $\mathbf{N}\,(E)$ is a distribution.
- Its values are in the same proportion as the values of E.

Normalisation

Definition (Normalisation)

If E's variables are n and m, then

$$\mathbf{N}\,(E) \;\widehat{=}\; E\,/\left(\sum n, m \bullet E\right)$$

- Let E be an expression:
  - Whose value (for all assignments of values) is nonnegative.
  - Whose sum (over all assignments of values) is strictly between 0 and ∞.
- Then, the normalisation $\mathbf{N}\,(E)$ is a distribution.
- Its values are in the same proportion as the values of E.

Normalisation

Definition (Normalisation)

If E's variables are n and m, then

$$\mathbf{N}\,(E) \;\widehat{=}\; E\,/\,\left(\sum n, m \bullet E\right)$$

- Let E be an expression:
  - Whose value (for all assignments of values) is nonnegative.
  - Whose sum (over all assignments of values) is strictly between 0 and ∞.
- Then, the normalisation $\mathbf{N}\,(E)$ is a distribution.
- Its values are in the same proportion as the values of E.

Normalisation

Definition (Normalisation)

If E's variables are n and m, then

$$\mathbf{N}\,(E) \;\widehat{=}\; E \,/\, \left(\sum n, m \bullet E\right)$$

- Let E be an expression:
  - Whose value (for all assignments of values) is nonnegative.
  - Whose sum (over all assignments of values) is strictly between 0 and ∞.
- Then, the normalisation $\mathbf{N}\,(E)$ is a distribution.
- Its values are in the same proportion as the values of E.

Normalisation

Definition (Normalisation)

If E's variables are n and m, then

$$\mathbf{N}\,(E) \;\widehat{=}\; E \,/\, \left(\sum n, m \bullet E\right)$$

- Let E be an expression:
  - Whose value (for all assignments of values) is nonnegative.
  - Whose sum (over all assignments of values) is strictly between 0 and ∞.
- Then, the normalisation $\mathbf{N}\,(E)$ is a distribution.
- Its values are in the same proportion as the values of E.

Normalisation

Definition (Normalisation)

If E's variables are n and m, then

$$\mathbf{N}\,(E) \;\widehat{=}\; E\,/\,(\sum n, m \bullet E)$$

- Let E be an expression:
  - Whose value (for all assignments of values) is nonnegative.
  - Whose sum (over all assignments of values) is strictly between 0 and ∞.
- Then, the normalisation $\mathbf{N}\,(E)$ is a distribution.
- Its values are in the same proportion as the values of E.

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $skip \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \,;\, B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0/x', y'] * B[x_0, y_0/x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $\mathit{skip} \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \;;\; B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0/x', y'] * B[x_0, y_0/x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $\mathit{skip} \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \;;\; B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0/x', y'] * B[x_0, y_0/x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $\mathit{skip} \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \;;\; B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0 / x', y'] * B[x_0, y_0 / x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $\mathit{skip} \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \,;\, B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0/x', y'] * B[x_0, y_0/x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $skip \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \;;\; B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0/x', y'] * B[x_0, y_0/x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $skip \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \;;\; B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0/x', y'] * B[x_0, y_0/x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

A Probabilistic Programming Language

- Iverson $[\,P\,] \;=\; (1 \lhd P \rhd 0) \;=\; \textbf{if } P \textbf{ then } 1 \textbf{ else } 0.$
- Inaction $skip \mathrel{\widehat{=}} [\,x' = x\,] * [\,y' = y\,]$
- Assignment $x := e \mathrel{\widehat{=}} [\,x' = e\,] * [\,y' = y\,]$
- Conditional $\textbf{if } c \textbf{ then } A \textbf{ else } B \mathrel{\widehat{=}} c * A + (1 - c) * B$
- Sequence $A \;;\; B \mathrel{\widehat{=}} \sum x_0, y_0 \bullet A[x_0, y_0/x', y'] * B[x_0, y_0/x, y]$
- Normalisation $A \parallel B \mathrel{\widehat{=}} \mathbf{N}\,(A * B)$
- Galois connection $\langle N \rangle_{\mathcal{I}} \sqsupseteq P \;=\; [\,N \leq [\,P\,]_{\mathcal{I}}\,]$

Iverson Laws

$$\langle N \rangle \;=\; N > 0$$

$$\langle 1 \rangle \;=\; \textbf{true}$$

$$\langle 0 \rangle \;=\; \textbf{false}$$

$$[\, N \;\leq\; [\langle N \rangle]\,]$$

$$\langle [P] \rangle \;\sqsupseteq\; P$$

$$[\neg \langle N \rangle] \;=\; [N = 0]$$

$$P \sqsupseteq Q \;\Rightarrow\; [P] \leq [Q]$$

$$M \leq N \;\Rightarrow\; \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[P \wedge Q] = [P] * [Q]$$

$$[P \vee Q] = [P] + [Q] - [P] * [Q]$$

$$[\neg P] = 1 - [P]$$

$$[k \in A] + [k \in B] = [k \in A \cup B] + [k \in A \cap B]$$

$$[x \in A \cap B] = [x \in A] * [x \in B]$$

$$[\forall m \bullet P(k,m)] = \prod m \bullet [P(k,m)]$$

$$[\exists m \bullet P(k,m)] = \min\{1, \sum m \bullet [P(k,m)]\}$$

$$\#\{\, m \mid P(k,m) \,\} = \sum m \bullet [P(k,m)]$$

Iverson Laws

$$\langle N \rangle \;=\; N > 0$$

$$\langle 1 \rangle \;=\; \textbf{true}$$

$$\langle 0 \rangle \;=\; \textbf{false}$$

$$[\, N \;\leq\; [\langle N \rangle]\,]$$

$$\langle [\,P\,] \rangle \;\sqsupseteq\; P$$

$$[\, \neg \langle N \rangle \,] \;=\; [\, N = 0 \,]$$

$$P \sqsupseteq Q \;\Rightarrow\; [\,P\,] \leq [\,Q\,]$$

$$M \leq N \;\Rightarrow\; \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\,P \wedge Q\,] = [\,P\,] * [\,Q\,]$$

$$[\,P \vee Q\,] = [\,P\,] + [\,Q\,] - [\,P\,] * [\,Q\,]$$

$$[\,\neg P\,] = 1 - [\,P\,]$$

$$[\,k \in A\,] + [\,k \in B\,] = [\,k \in A \cup B\,] + [\,k \in A \cap B\,]$$

$$[\,x \in A \cap B\,] = [\,x \in A\,] * [\,x \in B\,]$$

$$[\,\forall\, m \bullet P(k,m)\,] = \prod m \bullet [\,P(k,m)\,]$$

$$[\,\exists\, m \bullet P(k,m)\,] = \min\{1, \sum m \bullet [\,P(k,m)\,]\}$$

$$\#\{\, m \mid P(k,m) \,\} = \sum m \bullet [\,P(k,m)\,]$$

Iverson Laws

$$\langle N \rangle = N > 0$$

$$\langle 1 \rangle = \textbf{true}$$

$$\langle 0 \rangle = \textbf{false}$$

$$[\, N \leq [\langle N \rangle] \,]$$

$$\langle [P] \rangle \sqsupseteq P$$

$$[\, \neg \langle N \rangle \,] = [\, N = 0 \,]$$

$$P \sqsupseteq Q \Rightarrow [P] \leq [Q]$$

$$M \leq N \Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[P \wedge Q] = [P] * [Q]$$

$$[P \vee Q] = [P] + [Q] - [P] * [Q]$$

$$[\neg P] = 1 - [P]$$

$$[k \in A] + [k \in B] = [k \in A \cup B] + [k \in A \cap B]$$

$$[x \in A \cap B] = [x \in A] * [x \in B]$$

$$[\, \forall m \bullet P(k, m) \,] = \prod m \bullet [P(k, m)]$$

$$[\, \exists m \bullet P(k, m) \,] = \min\{1, \sum m \bullet [P(k, m)]\}$$

$$\#\{\, m \mid P(k, m) \,\} = \sum m \bullet [P(k, m)]$$

Iverson Laws

$$\langle N \rangle = N > 0$$

$$\langle 1 \rangle = \textbf{true}$$

$$\langle 0 \rangle = \textbf{false}$$

$$[\, N \leq [\,\langle N \rangle\,]\,]$$

$$\langle [\,P\,] \rangle \sqsupseteq P$$

$$[\, \neg \langle N \rangle \,] = [\, N = 0 \,]$$

$$P \sqsupseteq Q \Rightarrow [\,P\,] \leq [\,Q\,]$$

$$M \leq N \Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\,P \wedge Q\,] = [\,P\,] * [\,Q\,]$$

$$[\,P \vee Q\,] = [\,P\,] + [\,Q\,] - [\,P\,] * [\,Q\,]$$

$$[\,\neg P\,] = 1 - [\,P\,]$$

$$[\,k \in A\,] + [\,k \in B\,] = [\,k \in A \cup B\,] + [\,k \in A \cap B\,]$$

$$[\,x \in A \cap B\,] = [\,x \in A\,] * [\,x \in B\,]$$

$$[\,\forall\, m \bullet P(k,m)\,] = \prod m \bullet [\,P(k,m)\,]$$

$$[\,\exists\, m \bullet P(k,m)\,] = \min\{1, \sum m \bullet [\,P(k,m)\,]\}$$

$$\#\{\, m \mid P(k,m) \,\} = \sum m \bullet [\,P(k,m)\,]$$

Iverson Laws

$$\langle N \rangle = N > 0$$

$$\langle 1 \rangle = \textbf{true}$$

$$\langle 0 \rangle = \textbf{false}$$

$$[\, N \leq [\, \langle N \rangle \,] \,]$$

$$\langle [\, P \,] \rangle \sqsupseteq P$$

$$[\, \neg \langle N \rangle \,] = [\, N = 0 \,]$$

$$P \sqsupseteq Q \Rightarrow [\, P \,] \leq [\, Q \,]$$

$$M \leq N \Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\, P \wedge Q \,] = [\, P \,] * [\, Q \,]$$

$$[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,]$$

$$[\, \neg P \,] = 1 - [\, P \,]$$

$$[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,]$$

$$[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,]$$

$$[\, \forall\, m \bullet P(k, m) \,] = \prod m \bullet [\, P(k, m) \,]$$

$$[\, \exists\, m \bullet P(k, m) \,] = \min\{1, \sum m \bullet [\, P(k, m) \,]\}$$

$$\#\{\, m \mid P(k, m) \,\} = \sum m \bullet [\, P(k, m) \,]$$

Iverson Laws

$$\langle N\rangle = N > 0$$

$$\langle 1\rangle = \textbf{true}$$

$$\langle 0\rangle = \textbf{false}$$

$$[\,N \le [\langle N\rangle]\,]$$

$$\langle[P]\rangle \sqsupseteq P$$

$$[\,\neg\langle N\rangle\,] = [\,N = 0\,]$$

$$P \sqsupseteq Q \Rightarrow [P] \le [Q]$$

$$M \le N \Rightarrow \langle M\rangle \sqsupseteq \langle N\rangle$$

$$[P \wedge Q] = [P] * [Q]$$

$$[P \vee Q] = [P] + [Q] - [P] * [Q]$$

$$[\neg P] = 1 - [P]$$

$$[k \in A] + [k \in B] = [k \in A \cup B] + [k \in A \cap B]$$

$$[x \in A \cap B] = [x \in A] * [x \in B]$$

$$[\,\forall m \bullet P(k,m)\,] = \prod m \bullet [P(k,m)]$$

$$[\,\exists m \bullet P(k,m)\,] = \min\{1, \sum m \bullet [P(k,m)]\}$$

$$\#\{\,m \mid P(k,m)\,\} = \sum m \bullet [P(k,m)]$$

Iverson Laws

$$\langle N \rangle = N > 0$$

$$\langle 1 \rangle = \textbf{true}$$

$$\langle 0 \rangle = \textbf{false}$$

$$[\, N \le [\langle N \rangle]\,]$$

$$\langle [\, P \,] \rangle \sqsupseteq P$$

$$[\, \neg \langle N \rangle \,] = [\, N = 0 \,]$$

$$P \sqsupseteq Q \Rightarrow [\, P \,] \le [\, Q \,]$$

$$M \le N \Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\, P \wedge Q \,] = [\, P \,] * [\, Q \,]$$

$$[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,]$$

$$[\, \neg P \,] = 1 - [\, P \,]$$

$$[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,]$$

$$[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,]$$

$$[\, \forall m \bullet P(k,m) \,] = \prod m \bullet [\, P(k,m) \,]$$

$$[\, \exists m \bullet P(k,m) \,] = \min\{1, \sum m \bullet [\, P(k,m) \,]\}$$

$$\#\{\, m \mid P(k,m) \,\} = \sum m \bullet [\, P(k,m) \,]$$

Iverson Laws

$$\langle N \rangle \;=\; N > 0$$

$$\langle 1 \rangle \;=\; \textbf{true}$$

$$\langle 0 \rangle \;=\; \textbf{false}$$

$$[\, N \;\leq\; [\langle N \rangle]\,]$$

$$\langle [\, P \,] \rangle \;\sqsupseteq\; P$$

$$[\, \neg \langle N \rangle \,] \;=\; [\, N = 0 \,]$$

$$P \sqsupseteq Q \;\Rightarrow\; [\, P \,] \leq [\, Q \,]$$

$$M \leq N \;\Rightarrow\; \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\, P \wedge Q \,] = [\, P \,] * [\, Q \,]$$

$$[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,]$$

$$[\, \neg P \,] = 1 - [\, P \,]$$

$$[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,]$$

$$[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,]$$

$$[\, \forall\, m \bullet P(k,m) \,] = \prod m \bullet [\, P(k,m) \,]$$

$$[\, \exists\, m \bullet P(k,m) \,] = \min\{1, \sum m \bullet [\, P(k,m) \,]\}$$

$$\#\{\, m \mid P(k,m) \,\} = \sum m \bullet [\, P(k,m) \,]$$

Iverson Laws

$$\langle N \rangle = N > 0$$

$$\langle 1 \rangle = \textbf{true}$$

$$\langle 0 \rangle = \textbf{false}$$

$$[\, N \le [\langle N \rangle] \,]$$

$$\langle [\, P \,] \rangle \sqsupseteq P$$

$$[\, \neg \langle N \rangle \,] = [\, N = 0 \,]$$

$$P \sqsupseteq Q \Rightarrow [\, P \,] \le [\, Q \,]$$

$$M \le N \Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\, P \wedge Q \,] = [\, P \,] * [\, Q \,]$$

$$[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,]$$

$$[\, \neg P \,] = 1 - [\, P \,]$$

$$[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,]$$

$$[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,]$$

$$[\, \forall\, m \bullet P(k,m) \,] = \prod m \bullet [\, P(k,m) \,]$$

$$[\, \exists\, m \bullet P(k,m) \,] = \min\{1, \sum m \bullet [\, P(k,m) \,]\}$$

$$\#\{\, m \mid P(k,m) \,\} = \sum m \bullet [\, P(k,m) \,]$$

Iverson Laws

$$
\begin{aligned}
\langle N \rangle &= N > 0 \\
\langle 1 \rangle &= \textbf{true} \\
\langle 0 \rangle &= \textbf{false} \\
[\, N &\le [\langle N \rangle]\,] \\
\langle [P] \rangle &\sqsupseteq P \\
[\neg \langle N \rangle] &= [N = 0] \\
P \sqsupseteq Q &\Rightarrow [P] \le [Q] \\
M \le N &\Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle
\end{aligned}
$$

$$[P \wedge Q] = [P] * [Q]$$

$$[P \vee Q] = [P] + [Q] - [P] * [Q]$$

$$[\neg P] = 1 - [P]$$

$$[k \in A] + [k \in B] = [k \in A \cup B] + [k \in A \cap B]$$

$$[x \in A \cap B] = [x \in A] * [x \in B]$$

$$[\forall m \bullet P(k,m)] = \prod m \bullet [P(k,m)]$$

$$[\exists m \bullet P(k,m)] = \min\{1, \sum m \bullet [P(k,m)]\}$$

$$\#\{m \mid P(k,m)\} = \sum m \bullet [P(k,m)]$$

Iverson Laws

$$
\begin{aligned}
\langle N \rangle &= N > 0 \\
\langle 1 \rangle &= \textbf{true} \\
\langle 0 \rangle &= \textbf{false} \\
[\, N &\le [\langle N \rangle]\,] \\
\langle [P] \rangle &\sqsupseteq P \\
[\neg \langle N \rangle] &= [N = 0] \\
P \sqsupseteq Q &\Rightarrow [P] \le [Q] \\
M \le N &\Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle
\end{aligned}
$$

$$
\begin{aligned}
& [P \wedge Q] = [P] * [Q] \\
& [P \vee Q] = [P] + [Q] - [P] * [Q] \\
& [\neg P] = 1 - [P] \\
& [k \in A] + [k \in B] = [k \in A \cup B] + [k \in A \cap B] \\
& [x \in A \cap B] = [x \in A] * [x \in B] \\
& [\forall m \bullet P(k,m)] = \prod m \bullet [P(k,m)] \\
& [\exists m \bullet P(k,m)] = \min\{1, \sum m \bullet [P(k,m)]\} \\
& \#\{ m \mid P(k,m) \} = \sum m \bullet [P(k,m)]
\end{aligned}
$$

Iverson Laws

$$\langle N \rangle \;=\; N > 0$$

$$\langle 1 \rangle \;=\; \textbf{true}$$

$$\langle 0 \rangle \;=\; \textbf{false}$$

$$[\, N \;\leq\; [\langle N \rangle]\,]$$

$$\langle [\, P \,] \rangle \;\sqsupseteq\; P$$

$$[\, \neg \langle N \rangle \,] \;=\; [\, N = 0 \,]$$

$$P \sqsupseteq Q \;\Rightarrow\; [\, P \,] \leq [\, Q \,]$$

$$M \leq N \;\Rightarrow\; \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\, P \wedge Q \,] = [\, P \,] * [\, Q \,]$$

$$[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,]$$

$$[\, \neg\, P \,] = 1 - [\, P \,]$$

$$[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,]$$

$$[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,]$$

$$[\, \forall\, m \bullet P(k, m) \,] = \prod m \bullet [\, P(k, m) \,]$$

$$[\, \exists\, m \bullet P(k, m) \,] = \min\{1, \sum m \bullet [\, P(k, m) \,]\}$$

$$\#\{\, m \mid P(k, m) \,\} = \sum m \bullet [\, P(k, m) \,]$$

Iverson Laws

$$\langle N \rangle \;=\; N > 0$$

$$\langle 1 \rangle \;=\; \textbf{true}$$

$$\langle 0 \rangle \;=\; \textbf{false}$$

$$[\, N \;\leq\; [\langle N \rangle]\,]$$

$$\langle [P] \rangle \;\sqsupseteq\; P$$

$$[\neg \langle N \rangle] \;=\; [\, N = 0 \,]$$

$$P \sqsupseteq Q \;\Rightarrow\; [P] \leq [Q]$$

$$M \leq N \;\Rightarrow\; \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[P \wedge Q] = [P] * [Q]$$

$$[P \vee Q] = [P] + [Q] - [P] * [Q]$$

$$[\neg P] = 1 - [P]$$

$$[k \in A] + [k \in B] = [k \in A \cup B] + [k \in A \cap B]$$

$$[x \in A \cap B] = [x \in A] * [x \in B]$$

$$[\, \forall m \bullet P(k,m)\,] = \prod m \bullet [P(k,m)]$$

$$[\, \exists m \bullet P(k,m)\,] = \min\{1, \sum m \bullet [P(k,m)]\}$$

$$\#\{\, m \mid P(k,m)\,\} = \sum m \bullet [P(k,m)]$$

Iverson Laws

$$\langle N \rangle = N > 0$$

$$\langle 1 \rangle = \textbf{true}$$

$$\langle 0 \rangle = \textbf{false}$$

$$[\, N \leq [\, \langle N \rangle \,] \,]$$

$$\langle [\, P \,] \rangle \sqsupseteq P$$

$$[\, \neg \langle N \rangle \,] = [\, N = 0 \,]$$

$$P \sqsupseteq Q \Rightarrow [\, P \,] \leq [\, Q \,]$$

$$M \leq N \Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\, P \wedge Q \,] = [\, P \,] * [\, Q \,]$$

$$[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,]$$

$$[\, \neg P \,] = 1 - [\, P \,]$$

$$[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,]$$

$$[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,]$$

$$[\, \forall\, m \bullet P(k, m) \,] = \prod m \bullet [\, P(k, m) \,]$$

$$[\, \exists\, m \bullet P(k, m) \,] = \min\{1, \sum m \bullet [\, P(k, m) \,]\}$$

$$\#\{\, m \mid P(k, m) \,\} = \sum m \bullet [\, P(k, m) \,]$$

Iverson Laws

$$
\begin{aligned}
\langle N \rangle &= N > 0 \\
\langle 1 \rangle &= \textbf{true} \\
\langle 0 \rangle &= \textbf{false} \\
[\, N &\le [\langle N \rangle]\,] \\
\langle [\, P \,] \rangle &\sqsupseteq P \\
[\, \neg \langle N \rangle \,] &= [\, N = 0 \,] \\
P \sqsupseteq Q &\Rightarrow [\, P \,] \le [\, Q \,] \\
M \le N &\Rightarrow \langle M \rangle \sqsupseteq \langle N \rangle
\end{aligned}
$$

$$
\begin{aligned}
&[\, P \wedge Q \,] = [\, P \,] * [\, Q \,] \\
&[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,] \\
&[\, \neg\, P \,] = 1 - [\, P \,] \\
&[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,] \\
&[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,] \\
&[\, \forall\, m \bullet P(k, m) \,] = \prod m \bullet [\, P(k, m) \,] \\
&[\, \exists\, m \bullet P(k, m) \,] = \min\{1, \sum m \bullet [\, P(k, m) \,]\} \\
&\#\{\, m \mid P(k, m) \,\} = \sum m \bullet [\, P(k, m) \,]
\end{aligned}
$$

Iverson Laws

$$\langle N\rangle \;=\; N > 0$$

$$\langle 1\rangle \;=\; \textbf{true}$$

$$\langle 0\rangle \;=\; \textbf{false}$$

$$[\,N \;\le\; [\,\langle N\rangle\,]\,]$$

$$\langle[\,P\,]\rangle \;\sqsupseteq\; P$$

$$[\,\neg\,\langle N\rangle\,] \;=\; [\,N = 0\,]$$

$$P \sqsupseteq Q \;\Rightarrow\; [\,P\,] \le [\,Q\,]$$

$$M \le N \;\Rightarrow\; \langle M\rangle \sqsupseteq \langle N\rangle$$

$$[\,P \wedge Q\,] = [\,P\,] * [\,Q\,]$$

$$[\,P \vee Q\,] = [\,P\,] + [\,Q\,] - [\,P\,] * [\,Q\,]$$

$$[\,\neg\, P\,] = 1 - [\,P\,]$$

$$[\,k \in A\,] + [\,k \in B\,] = [\,k \in A \cup B\,] + [\,k \in A \cap B\,]$$

$$[\,x \in A \cap B\,] = [\,x \in A\,] * [\,x \in B\,]$$

$$[\,\forall\, m \bullet P(k,m)\,] = \prod m \bullet [\,P(k,m)\,]$$

$$[\,\exists\, m \bullet P(k,m)\,] = \min\{1, \sum m \bullet [\,P(k,m)\,]\}$$

$$\#\{\,m \mid P(k,m)\,\} = \sum m \bullet [\,P(k,m)\,]$$

Iverson Laws

$$\langle N \rangle \;=\; N > 0$$

$$\langle 1 \rangle \;=\; \textbf{true}$$

$$\langle 0 \rangle \;=\; \textbf{false}$$

$$[\, N \;\leq\; [\langle N \rangle]\,]$$

$$\langle [\, P \,] \rangle \;\sqsupseteq\; P$$

$$[\, \neg \langle N \rangle \,] \;=\; [\, N = 0 \,]$$

$$P \sqsupseteq Q \;\Rightarrow\; [\, P \,] \leq [\, Q \,]$$

$$M \leq N \;\Rightarrow\; \langle M \rangle \sqsupseteq \langle N \rangle$$

$$[\, P \wedge Q \,] = [\, P \,] * [\, Q \,]$$

$$[\, P \vee Q \,] = [\, P \,] + [\, Q \,] - [\, P \,] * [\, Q \,]$$

$$[\, \neg\, P \,] = 1 - [\, P \,]$$

$$[\, k \in A \,] + [\, k \in B \,] = [\, k \in A \cup B \,] + [\, k \in A \cap B \,]$$

$$[\, x \in A \cap B \,] = [\, x \in A \,] * [\, x \in B \,]$$

$$[\, \forall\, m \bullet P(k, m) \,] = \prod m \bullet [\, P(k, m) \,]$$

$$[\, \exists\, m \bullet P(k, m) \,] = \min\{1, \sum m \bullet [\, P(k, m) \,]\}$$

$$\#\{\, m \mid P(k, m) \,\} = \sum m \bullet [\, P(k, m) \,]$$

Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $1/2$ of success.
- *dalek* has probability $3/10$ of success.
- *cyber* attacks with probability of $3/5$.
- *dalek* attacks with probability of $2/5$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = 3/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = 1/2,$$
$$\mathrm{P}(\mathit{dalek}) = 2/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = 3/10$$

$$\begin{aligned}
& \mathrm{P}(\mathit{succ}) \\
= \; & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\
= \; & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\
= \; & 3/5 * 1/2 + 2/5 * 3/10 \; = \; 21/50
\end{aligned}$$

Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability ½ of success.
- *dalek* has probability $3/10$ of success.
- *cyber* attacks with probability of $3/5$.
- *dalek* attacks with probability of $2/5$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = 3/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = 1/2,$$
$$\mathrm{P}(\mathit{dalek}) = 2/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = 3/10$$

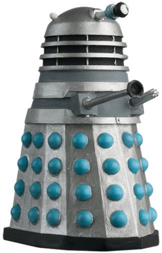
$$\begin{aligned}
&\mathrm{P}(\mathit{succ}) \\
={}& \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\
={}& \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\
={}& 3/5 * 1/2 + 2/5 * 3/10 = 21/50
\end{aligned}$$

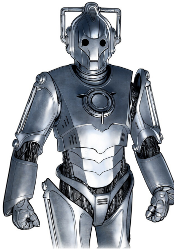
Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $^{1}/_{2}$ of success.
- *dalek* has probability $^{3}/_{10}$ of success.
- *cyber* attacks with probability of $^{3}/_{5}$.
- *dalek* attacks with probability of $^{2}/_{5}$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = {}^{3}/_{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = {}^{1}/_{2},$$
$$\mathrm{P}(\mathit{dalek}) = {}^{2}/_{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = {}^{3}/_{10}$$

$$\begin{aligned} & \mathrm{P}(\mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ = \; & {}^{3}/_{5} * {}^{1}/_{2} + {}^{2}/_{5} * {}^{3}/_{10} = {}^{21}/_{50} \end{aligned}$$





Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability ½ of success.
- *dalek* has probability $\frac{3}{10}$ of success.
- *cyber* attacks with probability of $\frac{3}{5}$.
- *dalek* attacks with probability of $\frac{2}{5}$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = \tfrac{3}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = \tfrac{1}{2},$$
$$\mathrm{P}(\mathit{dalek}) = \tfrac{2}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = \tfrac{3}{10}$$

$$\begin{aligned} & \mathrm{P}(\mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ = \; & \tfrac{3}{5} * \tfrac{1}{2} + \tfrac{2}{5} * \tfrac{3}{10} = \tfrac{21}{50} \end{aligned}$$

Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $1/2$ of success.
- *dalek* has probability $3/10$ of success.
- *cyber* attacks with probability of $3/5$.
- *dalek* attacks with probability of $2/5$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = 3/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = 1/2,$$
$$\mathrm{P}(\mathit{dalek}) = 2/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = 3/10$$

$$\begin{aligned} & \mathrm{P}(\mathit{succ}) \\ = {} & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ = {} & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ = {} & 3/5 * 1/2 + 2/5 * 3/10 = 21/50 \end{aligned}$$

Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $1/2$ of success.
- *dalek* has probability $3/10$ of success.
- *cyber* attacks with probability of $3/5$.
- *dalek* attacks with probability of $2/5$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

  $\mathrm{P}(\mathit{cyber}) = 3/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = 1/2,$
  $\mathrm{P}(\mathit{dalek}) = 2/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = 3/10$

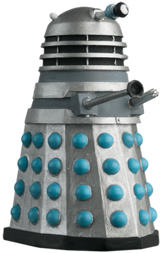
  $$\begin{aligned} & \mathrm{P}(\mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ = \; & 3/5 * 1/2 + 2/5 * 3/10 = 21/50 \end{aligned}$$

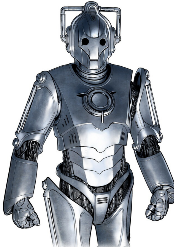
Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $\frac{1}{2}$ of success.
- *dalek* has probability $\frac{3}{10}$ of success.
- *cyber* attacks with probability of $\frac{3}{5}$.
- *dalek* attacks with probability of $\frac{2}{5}$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(cyber) = \frac{3}{5}, \quad \mathrm{P}(succ \mid cyber) = \frac{1}{2},$$
$$\mathrm{P}(dalek) = \frac{2}{5}, \quad \mathrm{P}(succ \mid dalek) = \frac{3}{10}$$

$$\begin{aligned} & \mathrm{P}(succ) \\ = \; & \mathrm{P}(cyber \wedge succ) + \mathrm{P}(dalek \wedge succ) \\ = \; & \mathrm{P}(cyber) * \mathrm{P}(succ \mid cyber) + \mathrm{P}(dalek) * \mathrm{P}(succ \mid dalek) \\ = \; & \frac{3}{5} * \frac{1}{2} + \frac{2}{5} * \frac{3}{10} = \frac{21}{50} \end{aligned}$$





Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability ½ of success.
- *dalek* has probability 3⁄10 of success.
- *cyber* attacks with probability of 3⁄5.
- *dalek* attacks with probability of 2⁄5.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = 3/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = 1/2,$$
$$\mathrm{P}(\mathit{dalek}) = 2/5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = 3/10$$

$$\begin{aligned} & \mathrm{P}(\mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ = \; & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ = \; & 3/5 * 1/2 + 2/5 * 3/10 = 21/50 \end{aligned}$$

Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability ½ of success.
- *dalek* has probability 3⁄10 of success.
- *cyber* attacks with probability of ⅗.
- *dalek* attacks with probability of ⅖.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = \tfrac{3}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = \tfrac{1}{2},$$

$$\mathrm{P}(\mathit{dalek}) = \tfrac{2}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = \tfrac{3}{10}$$

$$\begin{aligned} &\mathrm{P}(\mathit{succ}) \\ &= \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ &= \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ &= \tfrac{3}{5} * \tfrac{1}{2} + \tfrac{2}{5} * \tfrac{3}{10} = \tfrac{21}{50} \end{aligned}$$

Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $\frac{1}{2}$ of success.
- *dalek* has probability $\frac{3}{10}$ of success.
- *cyber* attacks with probability of $\frac{3}{5}$.
- *dalek* attacks with probability of $\frac{2}{5}$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = \frac{3}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = \frac{1}{2},$$
$$\mathrm{P}(\mathit{dalek}) = \frac{2}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = \frac{3}{10}$$

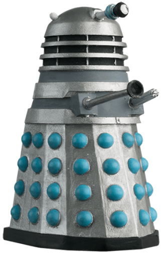
$$\begin{aligned} & \mathrm{P}(\mathit{succ}) \\ = {} & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ = {} & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ = {} & \frac{3}{5} * \frac{1}{2} + \frac{2}{5} * \frac{3}{10} = \frac{21}{50} \end{aligned}$$

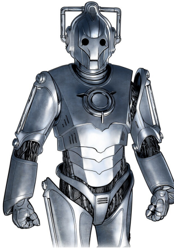
Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $\frac{1}{2}$ of success.
- *dalek* has probability $\frac{3}{10}$ of success.
- *cyber* attacks with probability of $\frac{3}{5}$.
- *dalek* attacks with probability of $\frac{2}{5}$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = \tfrac{3}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = \tfrac{1}{2},$$
$$\mathrm{P}(\mathit{dalek}) = \tfrac{2}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = \tfrac{3}{10}$$

$$\begin{aligned} &\mathrm{P}(\mathit{succ}) \\ &= \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ &= \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ &= \tfrac{3}{5} * \tfrac{1}{2} + \tfrac{2}{5} * \tfrac{3}{10} = \tfrac{21}{50} \end{aligned}$$





Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $\frac{1}{2}$ of success.
- *dalek* has probability $\frac{3}{10}$ of success.
- *cyber* attacks with probability of $\frac{3}{5}$.
- *dalek* attacks with probability of $\frac{2}{5}$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(cyber) = \frac{3}{5}, \quad \mathrm{P}(succ \mid cyber) = \frac{1}{2},$$
$$\mathrm{P}(dalek) = \frac{2}{5}, \quad \mathrm{P}(succ \mid dalek) = \frac{3}{10}$$

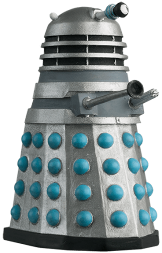
$$\begin{aligned} & \mathrm{P}(succ) \\ = \ & \mathrm{P}(cyber \wedge succ) + \mathrm{P}(dalek \wedge succ) \\ = \ & \mathrm{P}(cyber) * \mathrm{P}(succ \mid cyber) + \mathrm{P}(dalek) * \mathrm{P}(succ \mid dalek) \\ = \ & \frac{3}{5} * \frac{1}{2} + \frac{2}{5} * \frac{3}{10} = \frac{21}{50} \end{aligned}$$

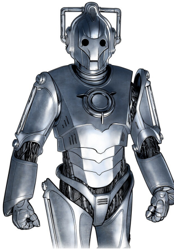
Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $\frac{1}{2}$ of success.
- *dalek* has probability $\frac{3}{10}$ of success.
- *cyber* attacks with probability of $\frac{3}{5}$.
- *dalek* attacks with probability of $\frac{2}{5}$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = \frac{3}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = \frac{1}{2},$$
$$\mathrm{P}(\mathit{dalek}) = \frac{2}{5}, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = \frac{3}{10}$$

$$\begin{aligned}
&\mathrm{P}(\mathit{succ}) \\
&= \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\
&= \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\
&= \frac{3}{5} * \frac{1}{2} + \frac{2}{5} * \frac{3}{10} = \frac{21}{50}
\end{aligned}$$



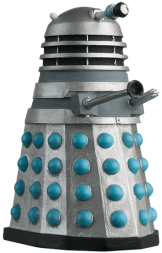


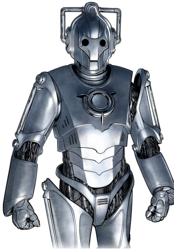
Example: Killer Robots

- cyberman and the dalek attack the Tardis daily.
- *cyber* has probability $^1\!/_2$ of success.
- *dalek* has probability $^3\!/_{10}$ of success.
- *cyber* attacks with probability of $^3\!/_5$.
- *dalek* attacks with probability of $^2\!/_5$.
- What is the probability of a successful attack?
- Conditional probability: $\mathrm{P}(A \wedge B) = \mathrm{P}(A) * \mathrm{P}(B \mid A)$.

$$\mathrm{P}(\mathit{cyber}) = {}^3\!/_5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = {}^1\!/_2,$$
$$\mathrm{P}(\mathit{dalek}) = {}^2\!/_5, \quad \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = {}^3\!/_{10}$$

$$\begin{aligned} & \mathrm{P}(\mathit{succ}) \\ = {} & \mathrm{P}(\mathit{cyber} \wedge \mathit{succ}) + \mathrm{P}(\mathit{dalek} \wedge \mathit{succ}) \\ = {} & \mathrm{P}(\mathit{cyber}) * \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) + \mathrm{P}(\mathit{dalek}) * \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) \\ = {} & {}^3\!/_5 * {}^1\!/_2 + {}^2\!/_5 * {}^3\!/_{10} = {}^{21}\!/_{50} \end{aligned}$$





Computational Approach

$\mathrm{P}(\mathit{cyber}) = {}^{3}\!/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = {}^{1}\!/_{2},$

$\mathrm{P}(\mathit{dalek}) = {}^{2}\!/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = {}^{3}\!/_{10}$

Computational Approach

$\mathrm{P}(\textit{cyber}) = {}^{3}\!/\!_{5}$, $\mathrm{P}(\textit{succ} \mid \textit{cyber}) = {}^{1}\!/\!_{2}$,
$\mathrm{P}(\textit{dalek}) = {}^{2}\!/\!_{5}$, $\mathrm{P}(\textit{succ} \mid \textit{dalek}) = {}^{3}\!/\!_{10}$

Computational Approach

$\mathrm{P}(\mathit{cyber}) = {}^{3}/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = {}^{1}/_{2},$

$\mathrm{P}(\mathit{dalek}) = {}^{2}/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = {}^{3}/_{10}$

Computational Approach

$\mathrm{P}(\mathit{cyber}) = {}^{3}\!/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = {}^{1}\!/_{2},$
$\mathrm{P}(\mathit{dalek}) = {}^{2}\!/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = {}^{3}\!/_{10}$

Computational Approach

$\mathrm{P}(\mathit{cyber}) = {}^{3}\!/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = {}^{1}\!/_{2},$
$\mathrm{P}(\mathit{dalek}) = {}^{2}\!/_{5}, \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = {}^{3}\!/_{10}$

Computational Approach

$\mathrm{P}(\mathit{cyber}) = \nicefrac{3}{5}, \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = \nicefrac{1}{2},$
$\mathrm{P}(\mathit{dalek}) = \nicefrac{2}{5}, \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = \nicefrac{3}{10}$

```
Tardis =
    if $\nicefrac{3}{5}$ then
        ( robot := cyber ;
          if $\nicefrac{1}{2}$ then attack := succ
          else attack := fail )
    else
        ( robot := dalek ;
          if $\nicefrac{3}{10}$ then attack := succ
          else attack := fail )
```

Computational Approach

$\mathrm{P}(\mathit{cyber}) = 3/5, \mathrm{P}(\mathit{succ} \mid \mathit{cyber}) = 1/2,$
$\mathrm{P}(\mathit{dalek}) = 2/5, \mathrm{P}(\mathit{succ} \mid \mathit{dalek}) = 3/10$

```
Tardis =
    if 3/5 then
        ( robot := cyber ;
          if 1/2 then attack := succ
          else attack := fail )
    else
        ( robot := dalek ;
          if 3/10 then attack := succ
          else attack := fail )
```

```
dtmc
const int cyber=1;
const int dalek=2;
const int succ=1;
const int fail=2;
module Tardis
   robot  : [1..2] init 1;
   attack : [1..2] init 1;
   s      : [0..3] init 0;
   [] s=0 -> 3/5: (robot'=cyber) & (s'=1)
           + 2/5: (robot'=dalek) & (s'=2);
   [] s=1 -> 1/2: (attack'=succ) & (s'=3)
           + 1/2: (attack'=fail) & (s'=3);
   [] s=2 -> 3/10:(attack'=succ) & (s'=3)
           + 7/10:(attack'=fail) & (s'=3);
   [] s=3 -> true;
endmodule
```

Computational Approach

$$\begin{aligned}
\mathit{Tardis} &= \textbf{if } \tfrac{3}{5} \textbf{ then } (\,\mathit{robot} := \mathrm{cyber}\ ;\ \textbf{if } \tfrac{1}{2} \textbf{ then } \mathit{attack} := \mathrm{succ} \textbf{ else } \mathit{attack} := \mathrm{fail}\,) \\
&\ \textbf{else } (\,\mathit{robot} := \mathrm{dalek}\ ;\ \textbf{if } \tfrac{3}{10} \textbf{ then } \mathit{attack} := \mathrm{succ} \textbf{ else } \mathit{attack} := \mathrm{fail}\,) \\
&= \tfrac{3}{10} * (\mathit{robot}, \mathit{attack} := \mathrm{cyber}, \mathrm{succ}) + \tfrac{3}{10} * (\mathit{robot}, \mathit{attack} := \mathrm{cyber}, \mathrm{fail}) \\
&\ + \tfrac{6}{50} * (\mathit{robot}, \mathit{attack} := \mathrm{dalek}, \mathrm{succ}) + \tfrac{14}{50} * (\mathit{robot}, \mathit{attack} := \mathrm{dalek}, \mathrm{fail})
\end{aligned}$$

- Probabilistic final states: assignments. Semantically equivalent to:
- Probability that $\mathit{attack}' = \mathrm{succ}$: $\tfrac{3}{10} + \tfrac{6}{50} = \tfrac{21}{50}$, the same answer as before.

Computational Approach

$$\begin{aligned}
\textit{Tardis} &= \textbf{if } \tfrac{3}{5} \textbf{ then } (\, \textit{robot} := \text{cyber} \;;\; \textbf{if } \tfrac{1}{2} \textbf{ then } \textit{attack} := \text{succ} \textbf{ else } \textit{attack} := \text{fail} \,) \\
&\quad\; \textbf{else } (\, \textit{robot} := \text{dalek} \;;\; \textbf{if } \tfrac{3}{10} \textbf{ then } \textit{attack} := \text{succ} \textbf{ else } \textit{attack} := \text{fail} \,) \\
&= \tfrac{3}{10} * (\textit{robot}, \textit{attack} := \text{cyber}, \text{succ}) + \tfrac{3}{10} * (\textit{robot}, \textit{attack} := \text{cyber}, \text{fail}) \\
&\quad + \tfrac{6}{50} * (\textit{robot}, \textit{attack} := \text{dalek}, \text{succ}) + \tfrac{14}{50} * (\textit{robot}, \textit{attack} := \text{dalek}, \text{fail})
\end{aligned}$$

- Probabilistic final states: assignments. Semantically equivalent to:
- Probability that $attack' = \text{succ}$: $\tfrac{3}{10} + \tfrac{6}{50} = \tfrac{21}{50}$, the same answer as before.

Computational Approach

$$
\begin{aligned}
\mathit{Tardis} &= \textbf{if } \tfrac{3}{5} \textbf{ then } (\, \mathit{robot} := \text{cyber} \;;\; \textbf{if } \tfrac{1}{2} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail} \,) \\
&\quad \textbf{else } (\, \mathit{robot} := \text{dalek} \;;\; \textbf{if } \tfrac{3}{10} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail} \,) \\
&= \tfrac{3}{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{succ}) + \tfrac{3}{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{fail}) \\
&\quad + \tfrac{6}{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{succ}) + \tfrac{14}{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{fail})
\end{aligned}
$$

- Probabilistic final states: assignments. Semantically equivalent to:
- Probability that $\mathit{attack}' = \text{succ}$: $\tfrac{3}{10} + \tfrac{6}{50} = \tfrac{21}{50}$, the same answer as before.

Computational Approach

$$
\begin{aligned}
\mathit{Tardis} \;&=\; \textbf{if } {}^{3}\!/\!_{5} \textbf{ then } (\, \mathit{robot} := \text{cyber} \;;\; \textbf{if } {}^{1}\!/\!_{2} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail} \,) \\
&\;\; \textbf{else } (\, \mathit{robot} := \text{dalek} \;;\; \textbf{if } {}^{3}\!/\!_{10} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail} \,) \\
&=\; {}^{3}\!/\!_{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{succ}) + {}^{3}\!/\!_{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{fail}) \\
&\;\; + {}^{6}\!/\!_{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{succ}) + {}^{14}\!/\!_{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{fail})
\end{aligned}
$$

- Probabilistic final states: assignments. Semantically equivalent to:
- Probability that $\mathit{attack}' = \text{succ}$: ${}^{3}\!/\!_{10} + {}^{6}\!/\!_{50} = {}^{21}\!/\!_{50}$, the same answer as before.

Computational Approach

$$\begin{aligned} \mathit{Tardis} &= \textbf{if } {}^{3}\!/_{5} \textbf{ then } (\, \mathit{robot} := \text{cyber} \;;\; \textbf{if } {}^{1}\!/_{2} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail}\,) \\ & \textbf{else } (\, \mathit{robot} := \text{dalek} \;;\; \textbf{if } {}^{3}\!/_{10} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail}\,) \\ &= {}^{3}\!/_{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{succ}) + {}^{3}\!/_{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{fail}) \\ &\quad + {}^{6}\!/_{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{succ}) + {}^{14}\!/_{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{fail}) \end{aligned}$$

- Probabilistic final states: assignments. Semantically equivalent to:
- Probability that $\mathit{attack}' = \text{succ}$: ${}^{3}\!/_{10} + {}^{6}\!/_{50} = {}^{21}\!/_{50}$, the same answer as before.

Computational Approach

$$\begin{aligned}
\textit{Tardis} &= \textbf{if } {}^{3}\!/_{5} \textbf{ then } (\, \textit{robot} := \text{cyber} \;;\; \textbf{if } {}^{1}\!/_{2} \textbf{ then } \textit{attack} := \text{succ} \textbf{ else } \textit{attack} := \text{fail} \,) \\
& \textbf{else } (\, \textit{robot} := \text{dalek} \;;\; \textbf{if } {}^{3}\!/_{10} \textbf{ then } \textit{attack} := \text{succ} \textbf{ else } \textit{attack} := \text{fail} \,) \\
&= {}^{3}\!/_{10} * (\textit{robot}, \textit{attack} := \text{cyber}, \text{succ}) + {}^{3}\!/_{10} * (\textit{robot}, \textit{attack} := \text{cyber}, \text{fail}) \\
& + {}^{6}\!/_{50} * (\textit{robot}, \textit{attack} := \text{dalek}, \text{succ}) + {}^{14}\!/_{50} * (\textit{robot}, \textit{attack} := \text{dalek}, \text{fail})
\end{aligned}$$

- Probabilistic final states: assignments. Semantically equivalent to:

```
module Tardis
  robot  : [1..2] init 1;
  attack : [1..2] init 1;
  t      : [0..1] init 0;
  [] t=0 -> 3/10:  (robot'=cyber) & (attack'=succ) & (t'=1)
            3/10:  (robot'=cyber) & (attack'=fail) & (t'=1)
            6/50:  (robot'=dalek) & (attack'=succ) & (t'=1)
            14/50: (robot'=dalek) & (attack'=fail) & (t'=1)
  [] t=3 -> true;
endmodule
```

- Probability that $\textit{attack}' = \text{succ}$: ${}^{3}\!/_{10} + {}^{6}\!/_{50} = {}^{21}\!/_{50}$, the same answer as before.

Computational Approach

$$\begin{aligned}
\mathit{Tardis} &= \textbf{if } {}^{3}\!/\!_{5} \textbf{ then } (\, \mathit{robot} := \text{cyber} \;;\; \textbf{if } {}^{1}\!/\!_{2} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail} \,) \\
&\ \textbf{else } (\, \mathit{robot} := \text{dalek} \;;\; \textbf{if } {}^{3}\!/\!_{10} \textbf{ then } \mathit{attack} := \text{succ} \textbf{ else } \mathit{attack} := \text{fail} \,) \\
&= {}^{3}\!/\!_{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{succ}) + {}^{3}\!/\!_{10} * (\mathit{robot}, \mathit{attack} := \text{cyber}, \text{fail}) \\
&\ + {}^{6}\!/\!_{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{succ}) + {}^{14}\!/\!_{50} * (\mathit{robot}, \mathit{attack} := \text{dalek}, \text{fail})
\end{aligned}$$

- Probabilistic final states: assignments. Semantically equivalent to:

```
module Tardis
  robot  : [1..2] init 1;
  attack : [1..2] init 1;
  t      : [0..1] init 0;
  [] t=0 -> 3/10:  (robot'=cyber) & (attack'=succ) & (t'=1)
            3/10:  (robot'=cyber) & (attack'=fail) & (t'=1)
            6/50:  (robot'=dalek) & (attack'=succ) & (t'=1)
            14/50: (robot'=dalek) & (attack'=fail) & (t'=1)
  [] t=3 -> true;
endmodule
```

- Probability that $\mathit{attack}' = \text{succ}$: ${}^{3}\!/\!_{10} + {}^{6}\!/\!_{50} = {}^{21}\!/\!_{50}$, the same answer as before.

Outline

Introduction

Motivation

Prism’s Semantics

Probabilistic Predicative Programming

Conclusions

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (1)

- Apply this semantics to unifying theories of uncertainty.
- Partially observable Markov decision processes, dynamic epistemic logic, . . .
- Research on describing and analysing uncertainty raises many questions.
- What would a unifying theory for uncertainty look like?
- What connects the semantics and tools that support different approaches?
- Can we establish more connections?
- Can we support probabilistic/statistical model checking with theorem proving?
- Can we support theorem proving with probabilistic/statistical model checking?
- Can we establish uncertainty properties using CbyC?

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.

Further Work (2)

- What about probabilistic refinement-based model checking?
- Can we qualify one target analysis tool for high assurance?
- What's the formal testing theory for a system with unknown MDP semantics?
- What are the testability hypotheses (in Gaudel's sense)?
- How do we exploit testing, proof, and model checking together?
- What about uncertainty modelling and runtime verification?
- How do we develop, apply, and evaluate uncertain systems?

- We have described preliminary work towards answering these questions.