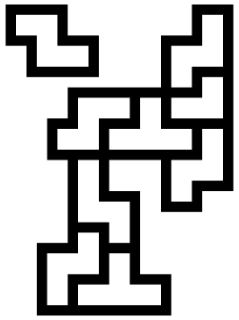
TU Graz
SCIENCE
PASSION
TECHNOLOGY

Course no. IND.04033UF (Lecture)
Course no. IND.04034UF (Practicals)

# Logic and Computability



Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at

March 1, 2024

# Outline

- Team
- Administrative Information
  - Lecture
  - Practicals
- Outline
- Teaser

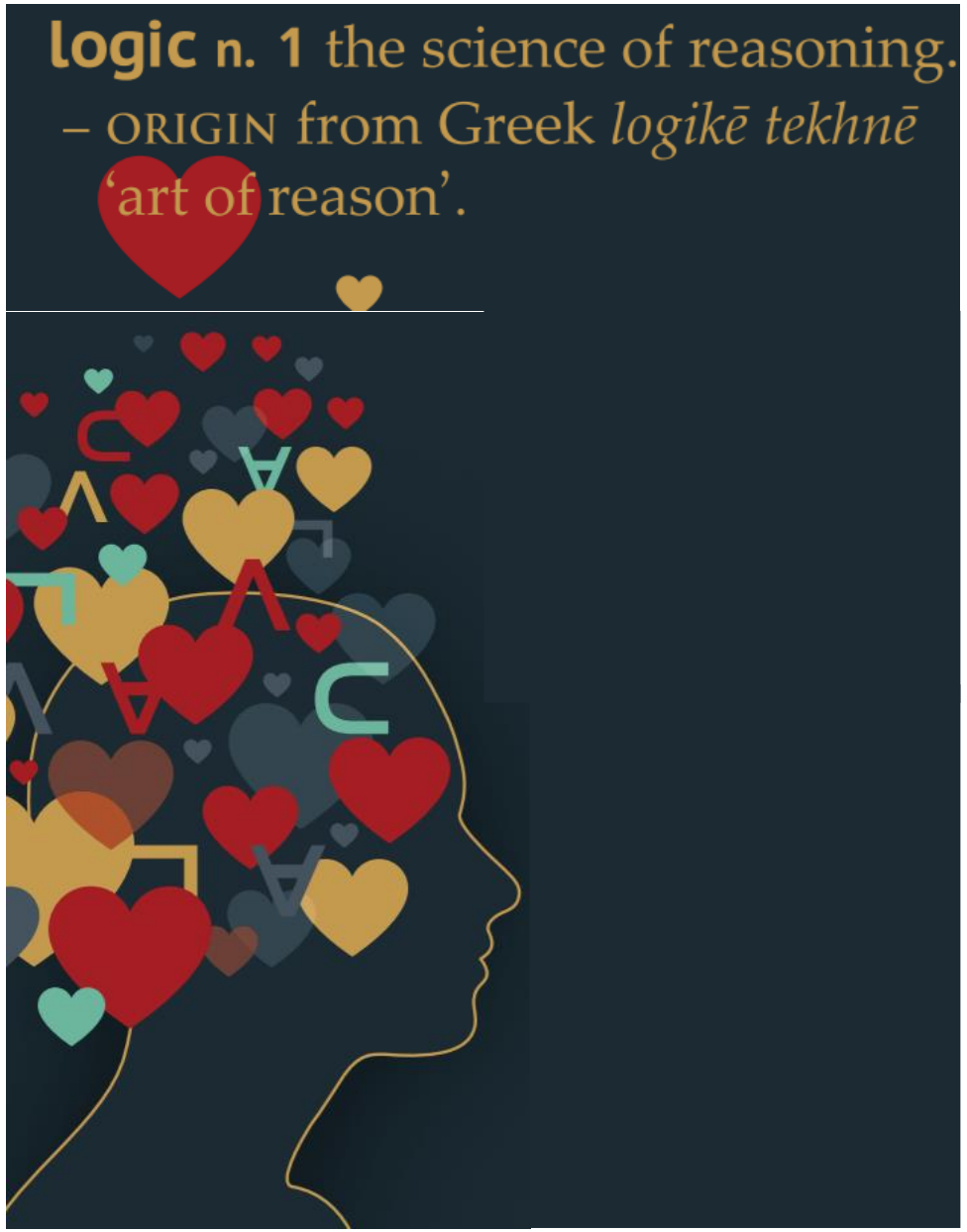



Image from Vienna Summer of Logic http://vsl2014.at/

# Bettina Könighofer

- Assistant Professor at IAIK
- Team: Trusted AI Group

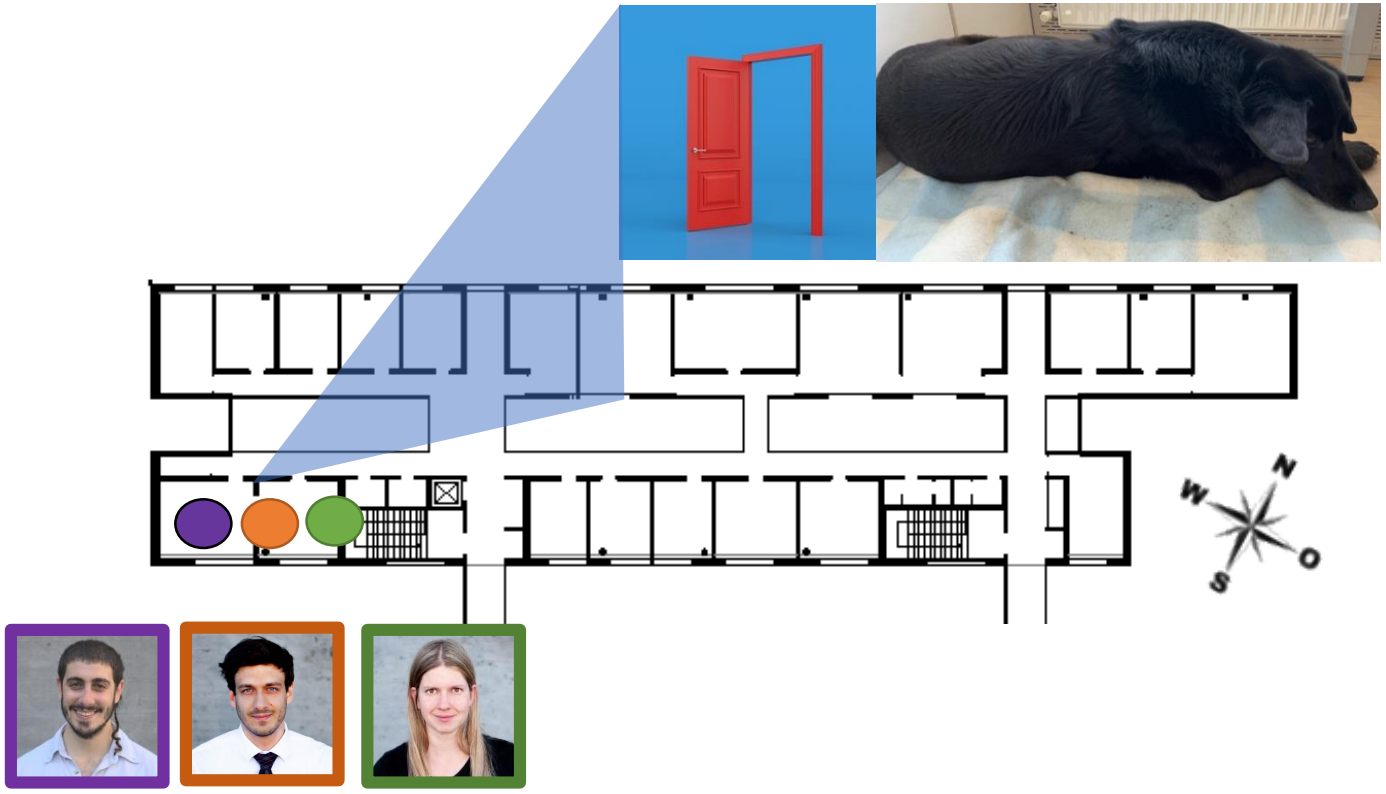
Filip Cano Cordoba

Stefan Pranger

- Teaching
  - Logic and Computability
  - Model Checking (CS Master)
  - ISW/Bachelor thesis/master project/master thesis (IAIK website)

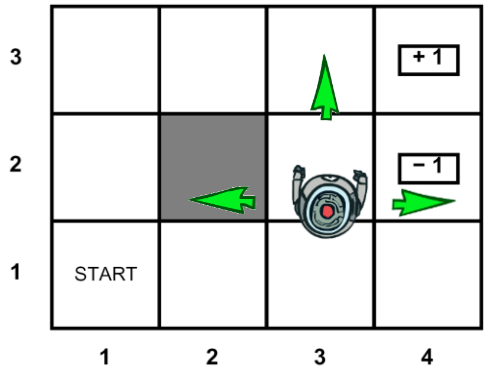
# Contact Details

- IAIK, Inffeldgasse 16a/II, Room IF02042
  - Open Door Policy
- 0316/873 – 5554
- bettina.koenighofer@iaik.tugraz.at
- stefan.pranger@iaik.tugraz.at
- Discord



# Bettina Könighofer

- Research:
  Combining **Symbolic AI** (model-based) and **Machine Learning** (model-free)


Model of environment

Learning Agent

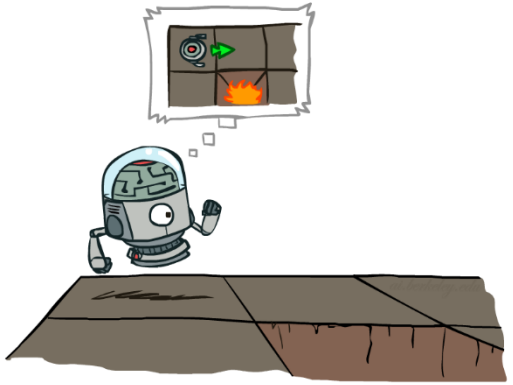
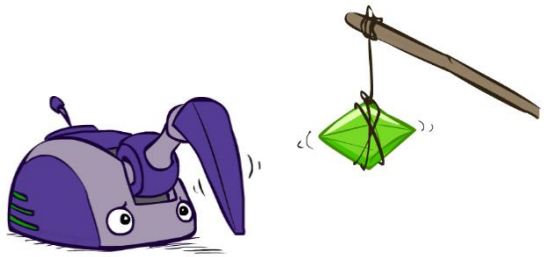


Image from http://ai.berkeley.edu/lecture_slides.html

# Bettina Könighofer

- Research:
  Combining **Symbolic AI** (model-based) and **Machine Learning** (model-free)

**Correctness Guarantees**

**Accountability**

**Explainability**

**Robustness**

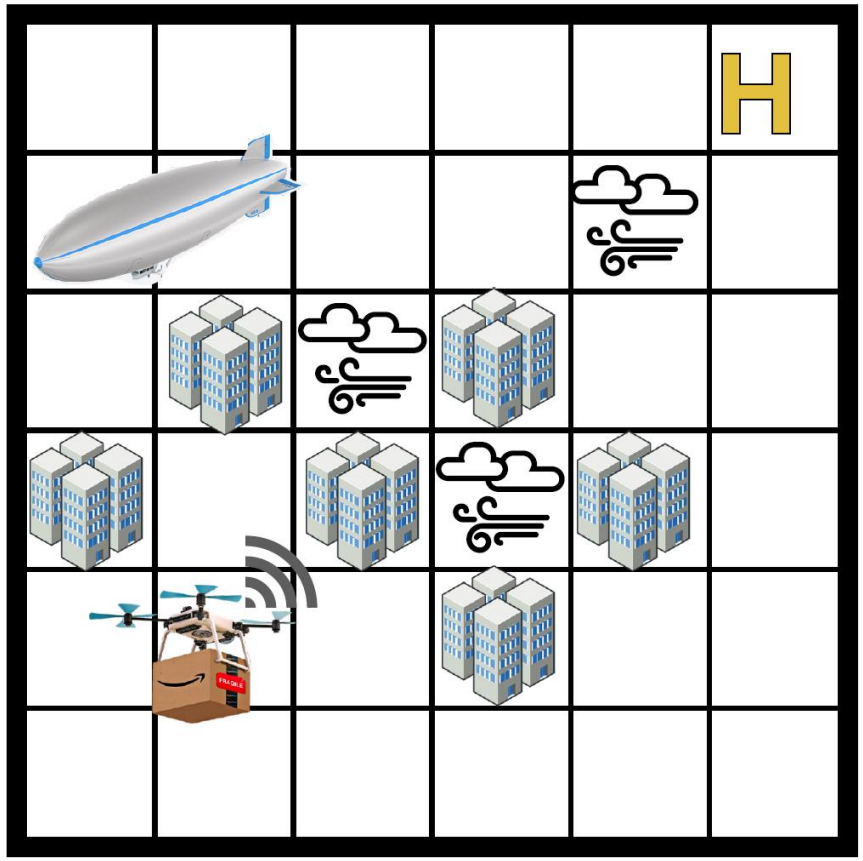
**Performance Guarantees**

**Fair Decision Making**

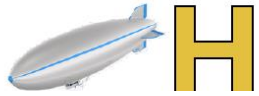
Image from http://ai.berkeley.edu/lecture_slides.html

# Shielded Reinforcement Learning

## Decision Making under Uncertainty




**Uncertainty** caused by sensor imprecision, wind gusts, and limited view



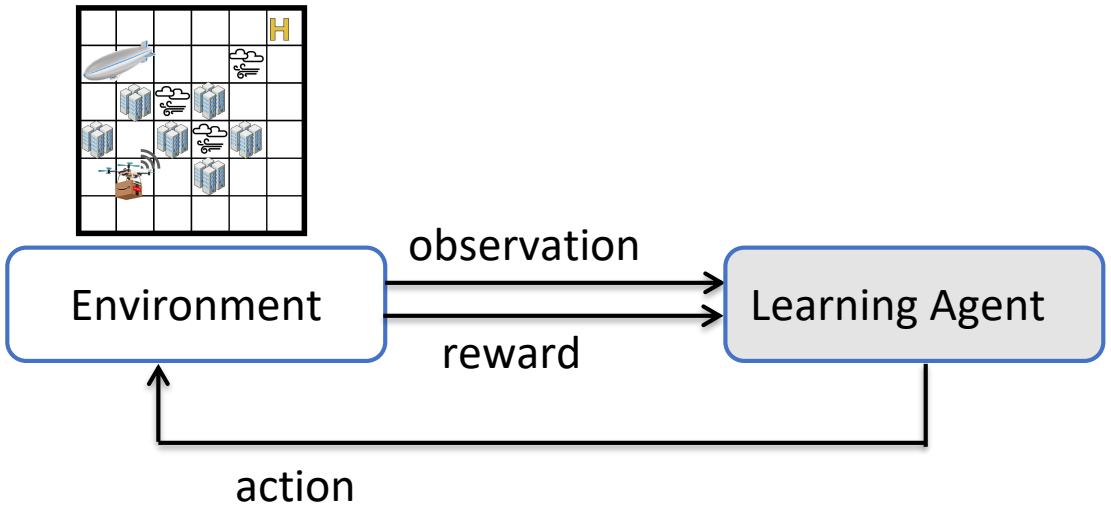

Complex **task specification**

N. Jansen, B. Könighofer, S. Junges, A. Serban, R. Bloem:
Safe Reinforcement Learning Using Probabilistic Shields. **CONCUR 2020**

# Reinforcement Learning

## RL agent…

- …explores environment by taking actions and observing feedback
- …finds optimal policy for making decisions




Find a policy $\pi$ that maximixes $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$

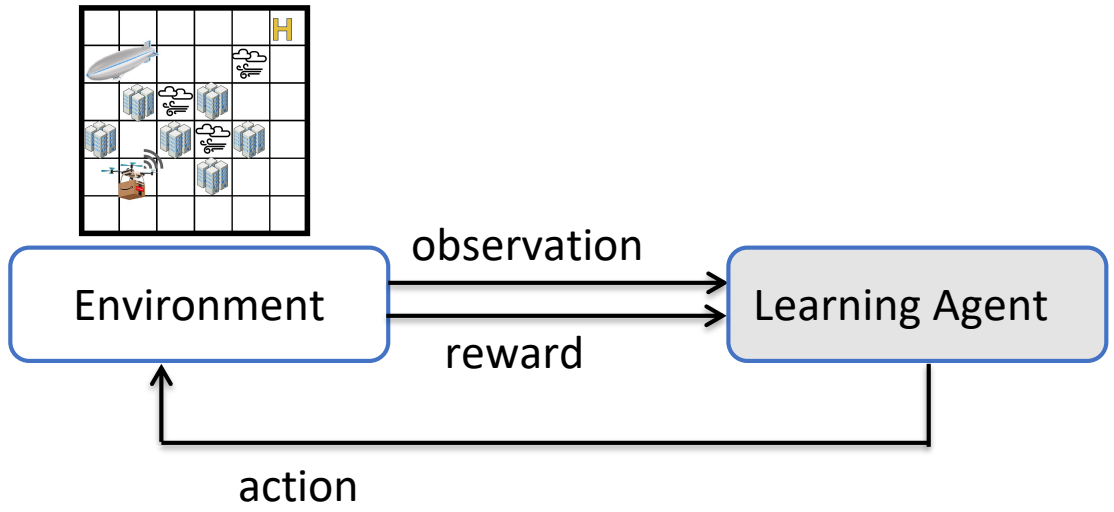
with the discount factor $0 \leq \gamma \leq 1$ and reward $R_t$ at time $t$

N. Jansen, B. Könighofer, S. Junges, A. Serban, R. Bloem:
Safe Reinforcement Learning Using Probabilistic Shields. **CONCUR 2020**

# Reinforcement Learning

## Limitations

- Exploration is **safety-critical**
- RL is **data-hungry**
- Rewards cannot capture **sophisticated task specifications**



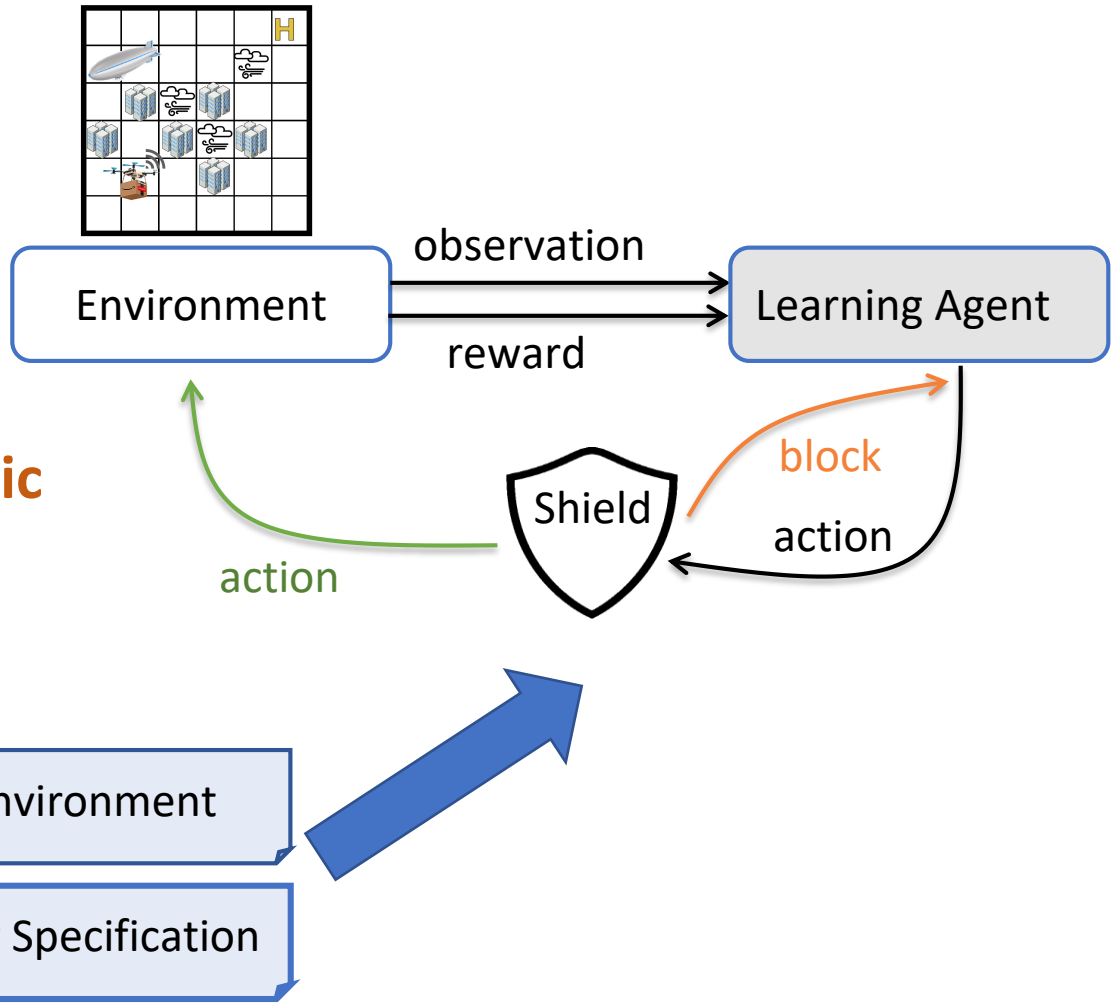

N. Jansen, B. Könighofer, S. Junges, A. Serban, R. Bloem:
Safe Reinforcement Learning Using Probabilistic Shields. **CONCUR 2020**

# Shielded Reinforcement Learning

„A crash can only occur with a **probability at most 0.001%**“

**Formal specification in probabilistic temporal logic**

$$Pr_{\leq 0.001}\big(Eventually(Crash)\big)$$

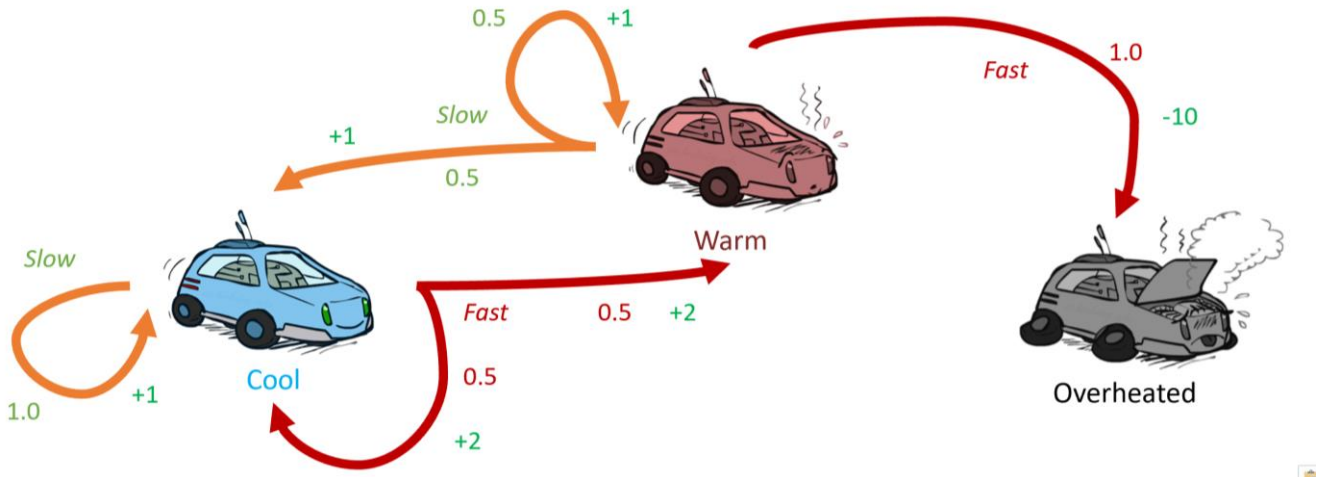


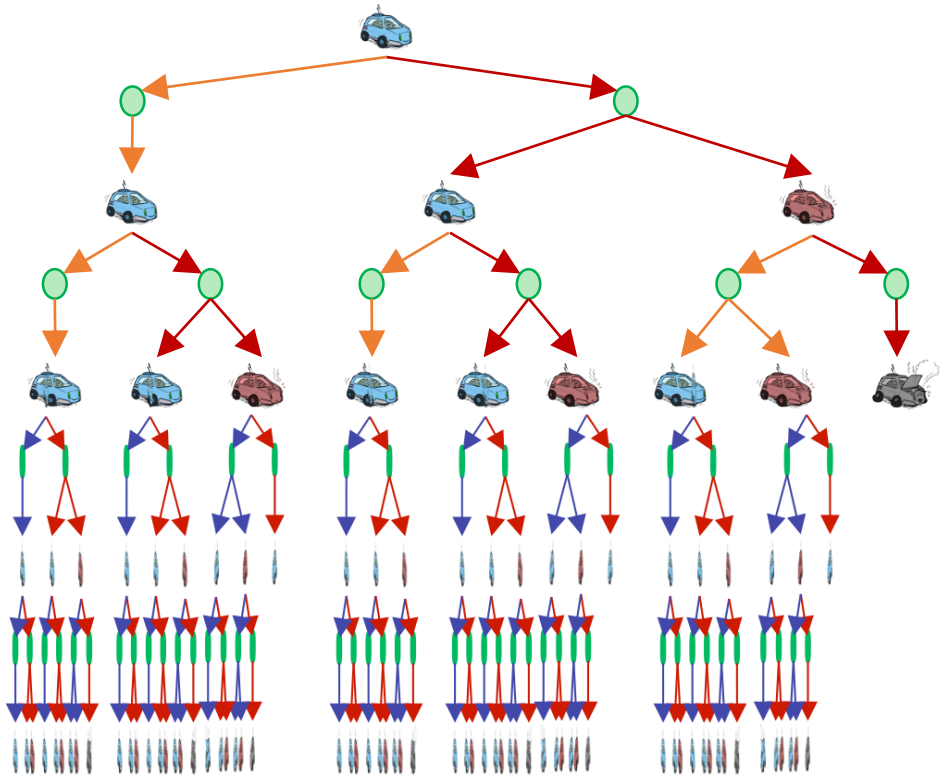

N. Jansen, B. Könighofer, S. Junges, A. Serban, R. Bloem:
Safe Reinforcement Learning Using Probabilistic Shields. **CONCUR 2020**

# Stefan Pranger

- University Assistant at IAIK

- Research
  - Safe Learning in Probabilistic Environments
  - Tool: TEMPEST
  - https://tempest-synthesis.org/



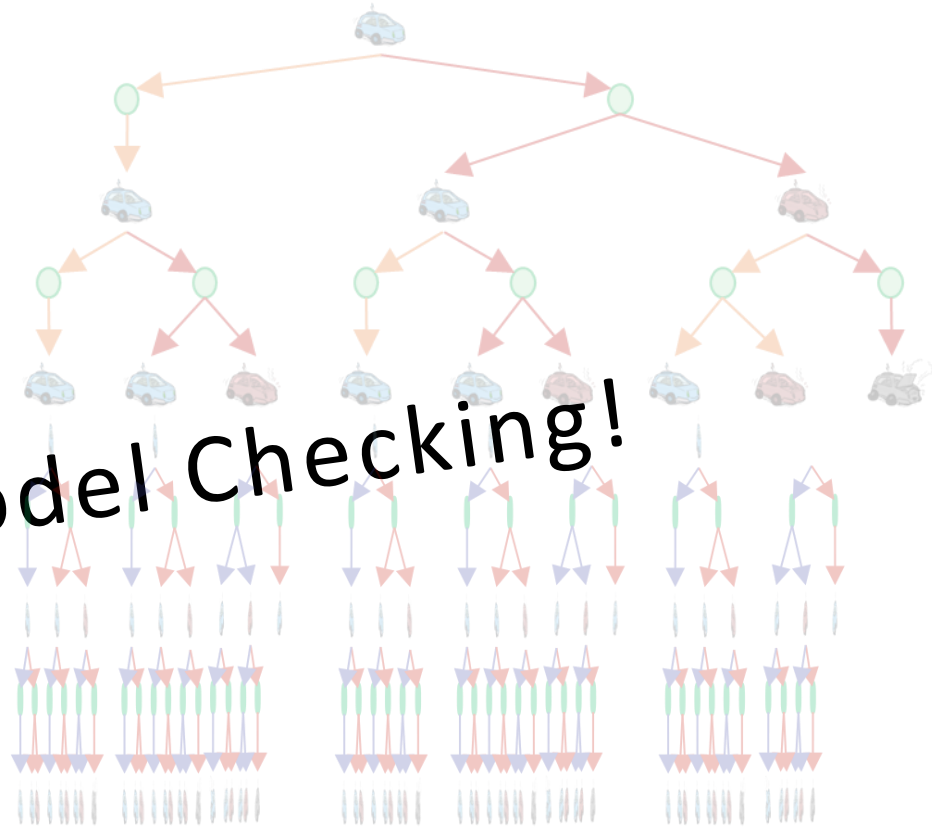



# Stefan Pranger

- University Assistant at IAIK

- Research
  - Safe Learning in Probabilistic Environments
  - Tool: TEMPEST
  - https://tempest-synthesis.org/


More about that in Model Checking!

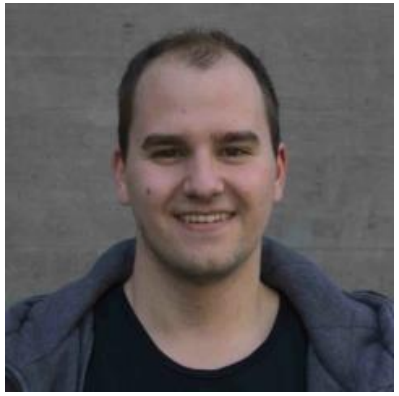


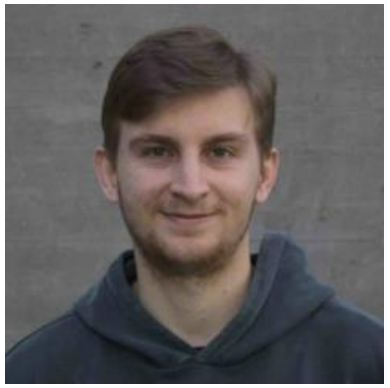
# Stefan Pranger

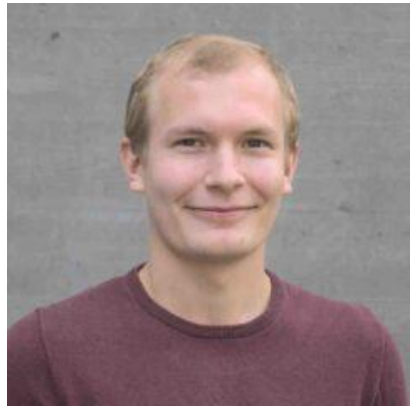
- University Assistant at IAIK
- Research
  - Safe Learning in Probabilistic Environments
  - Tool: TEMPEST
- Teaching
  - Logic and Computability
  - Model Checking
  - Bachelor thesis/master project/master thesis

# Teaching Assistants


- Arthur Lippitz
  - arthur.lippitz@student.tugraz.at
- Julian Rakuschek
  - julian.rakuschek@tugraz.at
- Paul Gollob
  - paul.gollob@iaik.tugraz.at
- Verena Schaffer
  - verena.schaffer@student.tugraz.at
- Matthias Grilz
  - matthias.grilz@student.tugraz.at

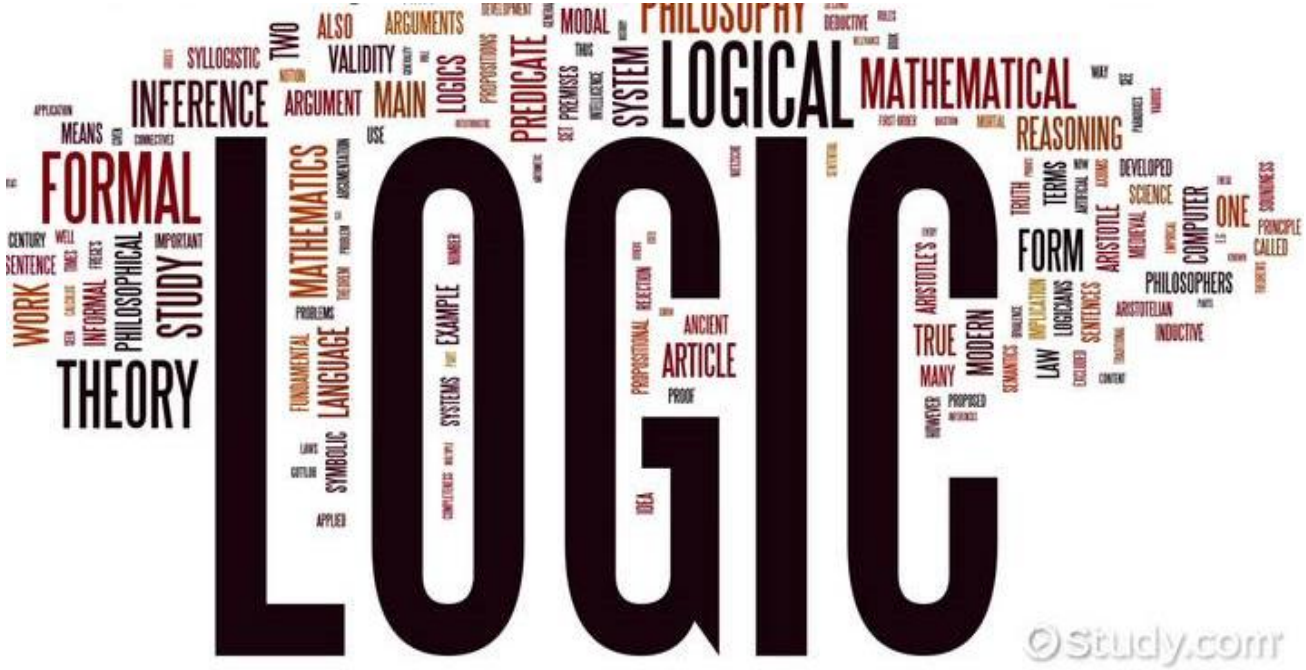





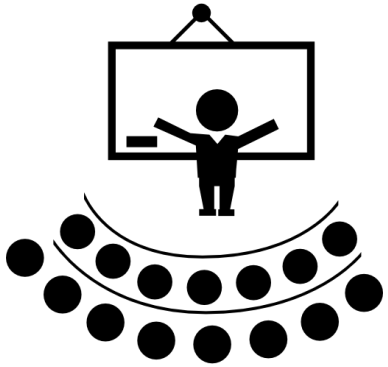


# Outline

- Team
- **Administrative Information**
  - **Lecture**
  - Practicals
- Outline
- Teaser



# Lecture



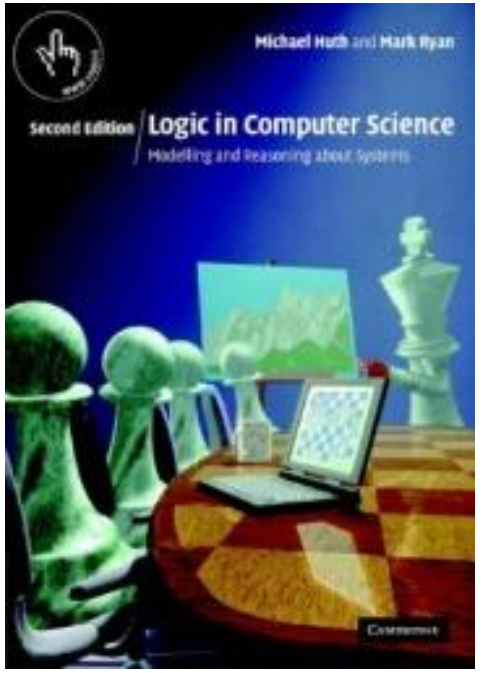
- Friday 10:15am-11:45am, HS i13
- Very interactive
- Solve examples together
  - Bring pen and paper / tablet / coffee
  - Why:
    - Self-control
    - Apply new knowledge immediately

# Material

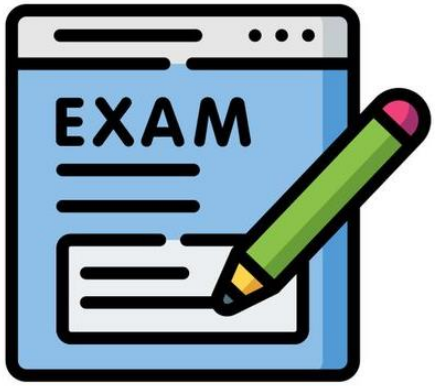
- Course website
  - https://www.iaik.tugraz.at/lc
- Material
  - Slides
  - Lecture Recordings
  - Lecture notes
  - Questionnaire
    - Exam questions (+ solutions)

# Material

- Course website
  - https://www.iaik.tugraz.at/lc
- Material
  - Slides
  - Lecture Recordings
  - Lecture notes
  - Questionnaire
  - Book
    - Huth and Ryan,
      Logic in Computer Science,
      Cambridge University Press, 2004

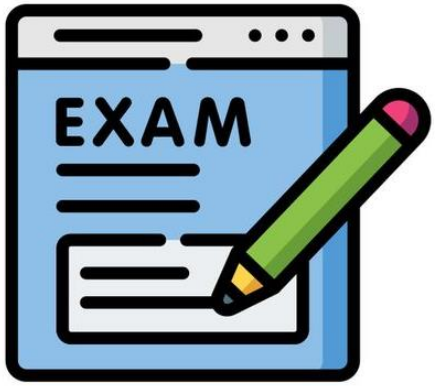


# Exam




- Consists only of questions from **questionnaire**
- We will solve examples from questionnaire during class.
- Assignments 1-5 consist of questions from questionnaire.
- You prepare for the exam during
  - the lecture, and
  - the practicals.

# Exam

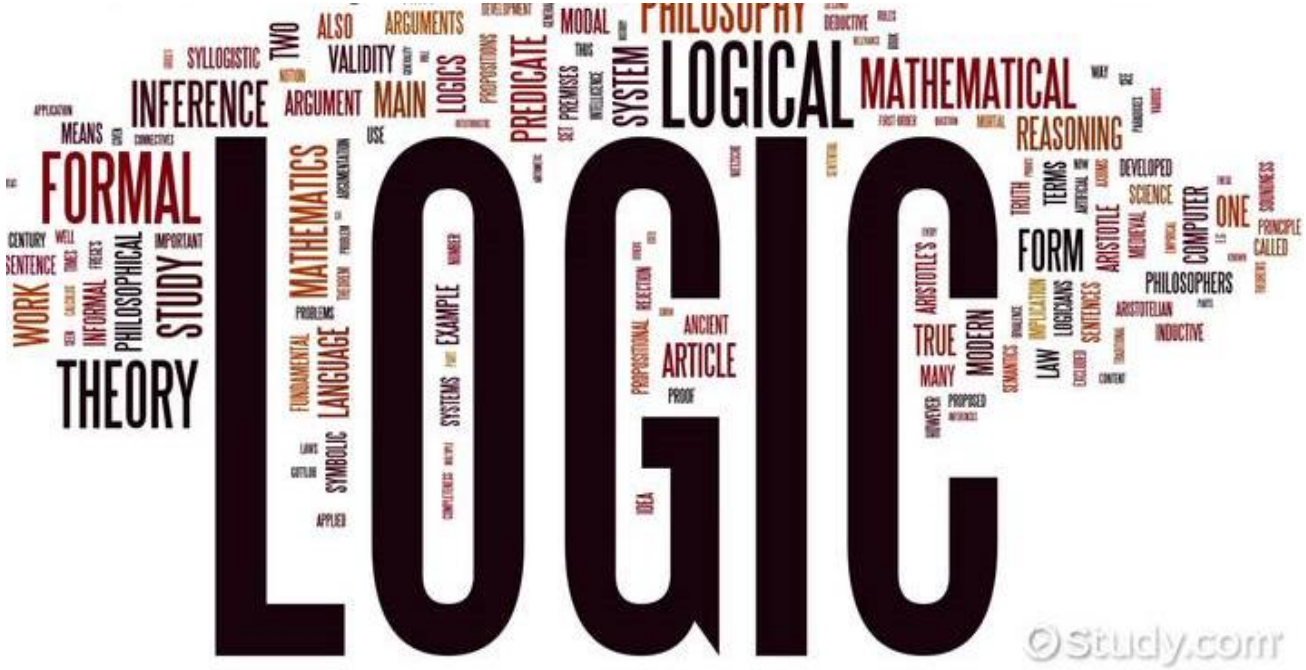



- Written exam at the end of the semester: 21.06.2024
- Question hour (Training exam): 14.06.2024
- **Voluntary training evening**
  - 19.06.2023  4pm - open end (Wednesday)
  - Students can study for exam. We are there to help.

# Outline

- Team
- Administrative Information
  - Lecture
  - **Practicals**

- Outline
- Teaser



# Assignments

- 6 Assignments
  - 5 pen-and-paper assignment sheets
  - 1 programming assignment sheets

| Number | Topic | Kick-Off | Deadline |
|---|---|---|---|
| 1 | SAT Solving + Binary Decision Diagrams | 2024-03-08 | 2024-03-20 |
| 2 | Natural Deduction for Propositional Logic | 2024-03-22 | 2024-04-10 |
| 3 | Equivalence Checking + Predicate Logic | 2024-04-12 | 2024-04-24 |
| 4 | Natural Deduction for Predicate Logic | 2024-05-03 | 2024-05-15 |
| 5 | Satisfiability Modulo Theory | 2024-05-17 | 2024-05-22 |
| 6 | Programming Assignment (Z3) | TBA | 2024-06-05 |

# Assignments

- Assignment 1-5 – Pen & Paper
  - Tick via TeachCenter
  - Deadline: Wednesday 11:59 pm
  - Present in class

# Practical classes

- Attendance is compulsory
  - Discussion of Pen & Paper exercises
  - Replacement interview 1 week later
    - Thursday: 1pm, IAIK, Inffeldgasse 16a, 2nd floor
- Students present solutions
- Inability to explain solution or completely wrong solutions lead to point deduction
  - Either 50% or 100% of assignment
  - Minor errors are OK!

# Assignments

- Assignment 1-5 – Pen & Paper
  - Tick via TeachCenter
  - Deadline: Wednesday 11:59 pm
  - Present in class
- Assignment 6 – Programming
  - Groups of 2 students
  - Programming exercises handled via git
  - Individual interviews per group

# Grading

- Assignment 1-5: 15 points
- Assignment 6: 25 points

- If Points…
  - $\geq$ 87.5: (1) Sehr Gut / Excellent
  - $\geq$ 75.0: (2) Gut / Good
  - $\geq$ 62.5: (3) Befriedigend / Satisfactory
  - $\geq$ 50.0: (4) Genügend / Sufficient
  - $<$ 50.0: (5) Nicht Genügend / Insufficient

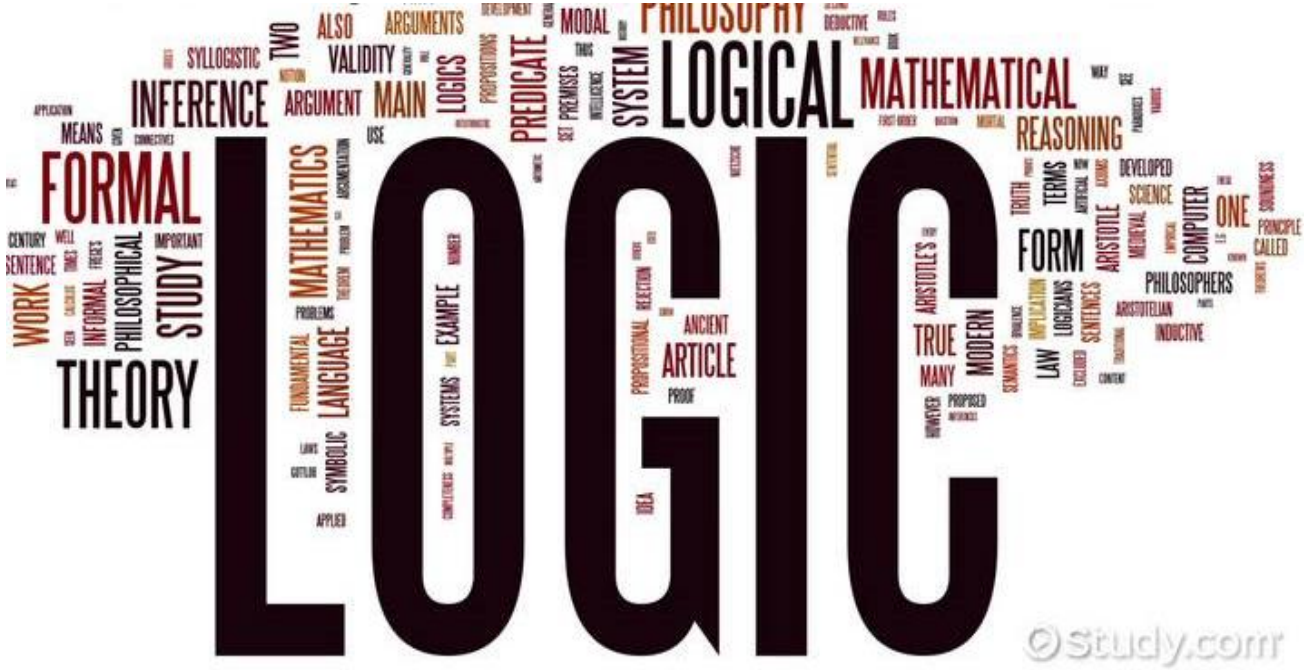
# Communication

- Discord Server
- E-Mail
  - bettina.koenighofer@iaik.tugraz.at
  - stefan.pranger@iaik.tugraz.at
- Visit us at IAIK – Open door policy

# Outline

- Team
- Administrative Information
  - Lecture
  - Practicals

- Outline
- Teaser



# Time Line - Topics

**Lectures 1 – 4:**
**Propositional Logic**

**Lectures 5 – 9:**
**Predicate Logic**

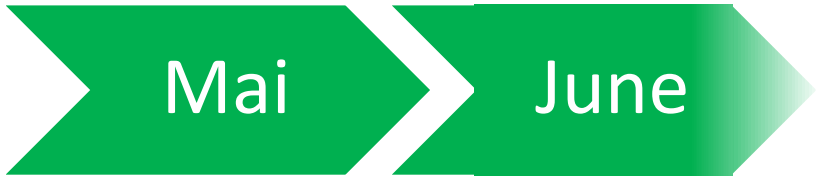

**Lecture 10: Temporal Logic**
**Lecture 11: Decidability**

# Propositional Logic

March April

- **Syntax & Semantic**
  - How do formulate problems
- **Algorithms to decide satisfiability**
  - Deciding propositional formulas with DPLL (with CDCL)
- **Data structures**
  - Binary Decision Diagrams (BDDs)
- **Natural deduction**
  - Perform proofs
- **Equivalence checking and normal forms**

# Predicate Logic

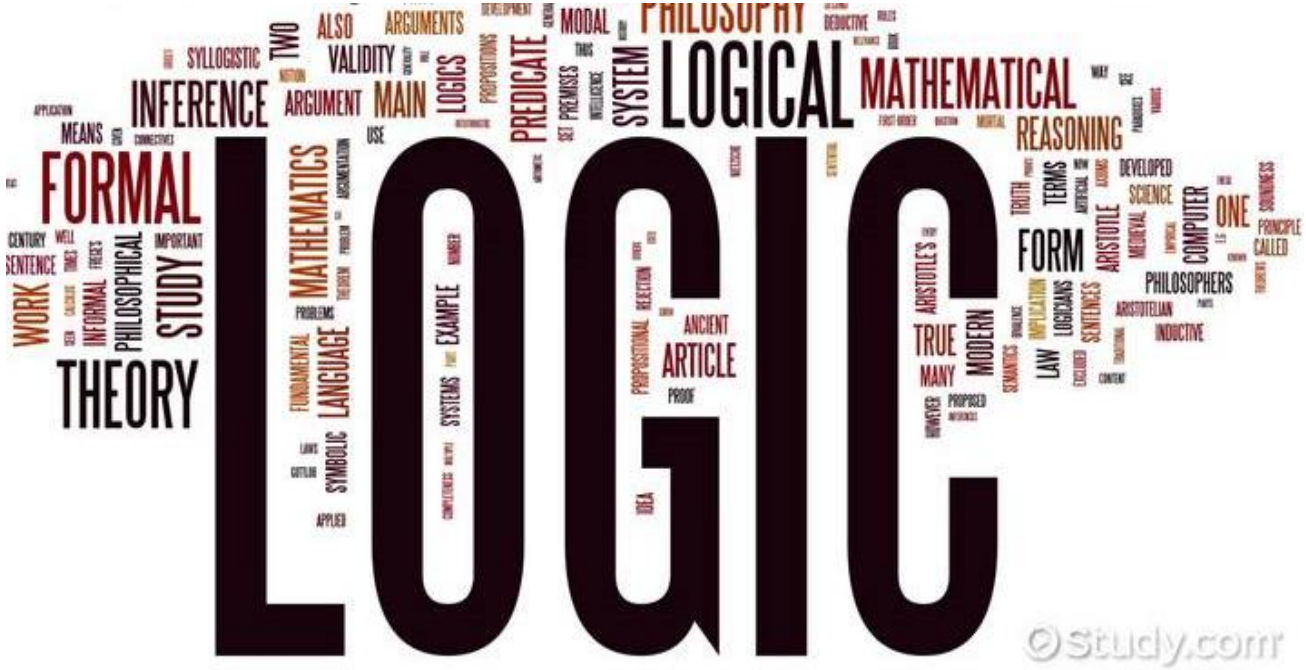



- **Syntax & Semantic**
- **Natural deduction**
  - Perform proofs
- **Satisfiability Modulo Theory (SMT)**
  - Formulas in predicate logic with theories
- **Algorithms to decide satisfiability**
  - Deciding SMT formulas (Eager encoding and DPLL(T))
- **SMT in Practice - Z3**

# Outline

- Team
- Administrative Information
  - Lecture
  - Practicals

- Outline
- Teaser



# Translate Sentences to Formulas

- *"I like Fridays and I don't like Mondays."*

Sentence that can be true or false

p… I like Fridays

Sentence that can be true or false

q… I like Mondays

$p \wedge \neg q$

**Logical Operators**

$\wedge \cdots AND$
$\vee \ \ldots OR$
$\neg \ldots NOT$
$\rightarrow \cdots IMPLICATION$

# Translate the Sentences to Formulas

- *"If today is Friday, then tomorrow is Saturday."*

  $p$... today is Friday, q… tomorrow is Saturday

  $$p \rightarrow q$$

- *"This lecture is exciting and not boring."*

  $p$... This lecture is exciting, q… This lecture is boring

  $$p \wedge \neg q$$

**Logical Operators**

$\wedge \cdots AND$
$\vee \ \ldots OR$
$\neg \ldots NOT$
$\rightarrow \cdots IMPLICATION$

# Quiz – Translate the Sentences to Formulas

- You can fool some people sometimes
- You can fool some of the people all the time
- You can fool some people sometimes but you can‘t fool all the people all the time [Bob Marley]
- You can fool some of the people all of the time, and all of the people some of the time, but you cannot fool all of the people all of the time [Abraham Lincoln]

# A Solution….

$Fool(p, t)$ … returns True if you can fool person p at time t

$\exists x: \varphi$ … returns true if there exists an x that makes $\varphi$ true

- You can fool some people sometimes

  $\exists p \in people\ \exists t \in time: Fool(p, t)$

- You can fool some of the people all the time

# A Solution….

$Fool(p, t)$ … returns True if you can fool person p at time t

$\exists x: \varphi$ … returns true if there exists an x that makes $\varphi$ true
$\forall x: \varphi$ … returns true if forall x that makes $\varphi$ true

- You can fool some people sometimes

$$\exists p \in people\ \exists t \in time: Fool(p, t)$$

- You can fool some of the people all the time

$$\exists p \in people\ \forall t \in time: Fool(p, t)$$

# A Solution….

$Fool(p, t)$ … returns True if you can fool person p at time t

$\exists x: \varphi$ … returns true if there exists an x that makes $\varphi$ true
$\forall x: \varphi$ … returns true if forall x that makes $\varphi$ true

- You can fool some people sometimes but you can‘t fool all the people all the time [Bob Marley]

$$\exists p \in people\ \exists t \in time: Fool(p, t)$$

# A Solution….

$Fool(p, t)$ … returns True if you can fool person p at time t

$\exists x: \varphi$ … returns true if there exists an x that makes $\varphi$ true
$\forall x: \varphi$ … returns true if forall x that makes $\varphi$ true

- You can fool some people sometimes but you can‘t fool all the people all the time [Bob Marley]

$$(\exists p \in people\ \exists t \in time: Fool(p, t)) \wedge \neg(\forall x \in people\ \forall t \in time: Fool(p, t))$$

# A Solution….

$Fool(p, t)$ … returns True if you can fool person p at time t

$\exists x: \varphi$ … returns true if there exists an x that makes $\varphi$ true
$\forall x: \varphi$ … returns true if forall x that makes $\varphi$ true

- You can fool some of the people all of the time,
  and all of the people some of the time,
  but you cannot fool all of the people all of the time [Abraham Lincoln]

$$(\exists p \in people\ \forall t \in time: Fool(p, t)) \wedge$$
$$\big(\forall p \in people\ \exists t \in time: Fool(p, t)\big) \wedge$$
$$\neg(\forall p \in people\ \forall t \in time: Fool(p, t))$$

# A Solution….

$Fool(p, t)$ … returns True if you can fool person p at time t

$\exists x: \varphi$ … returns true if there exists an x that makes $\varphi$ true
$\forall x: \varphi$ … returns true if forall x that makes $\varphi$ true

*Now you know some basics of predicate logic ☺*

# Quiz 2 - Translate the Sentences to Formulas

- *"Always, if there is a request, then there is a grant in the next step."*
- *"$grant_1$ and a $grant_2$ are never allowed simultaneously."*
- *"Always, a request will be granted in the next 3 time steps"*
- *"Any request will be granted eventually"*

**Temporal Operators**

G ... $Globally, Always$

F ... $Finally, Eventually$

X ... $Next$

# Quiz 2 - Translate the Sentences to Formulas

- *"Always, if there is a request, then there is a grant in the next step."*

$p$... there is a request, q… there is a grant

$$G(p \rightarrow Xq)$$

**Temporal Operators**

G ... $Globally, Always$
F ... $Finally, Eventually$
X ... $Next$

# Quiz 2 - Translate the Sentences to Formulas

- *"$grant_1$ and a $grant_2$ are never allowed simultaneously."*

$p$... $grant_1$ is allowed, q… $grant_2$ is allowed

$$G\neg(p \wedge q)$$

***Temporal Operators***

G ... $Globally, Always$

F ... $Finally, Eventually$

X ... $Next$

# Quiz 2 - Translate the Sentences to Formulas

- *"Always, a request will be granted in the next 3 time steps"*

$p$... there is a request, q… there is a grant

$$G(p \rightarrow XXXq)$$

**Temporal Operators**

G ... $Globally, Always$
F ... $Finally, Eventually$
X ... $Next$

# Quiz 2 - Translate the Sentences to Formulas

- *"Any request is granted eventually"*

$p$... the request is granted

$$GFp$$

**Temporal Operators**

G ... $Globally, Always$
F ... $Finally, Eventually$
X ... $Next$

# Quiz 2 - Translate the Sentences to Formulas

*Now you know some basics of temporal logic ☺*

**Temporal Operators**

G … $Globally, Always$
F … $Finally, Eventually$
X … $Next$

# Teaser - SMT

- SMT solvers are **magic**!
- You describe your problem (with a bit of code), the solver finds the answer

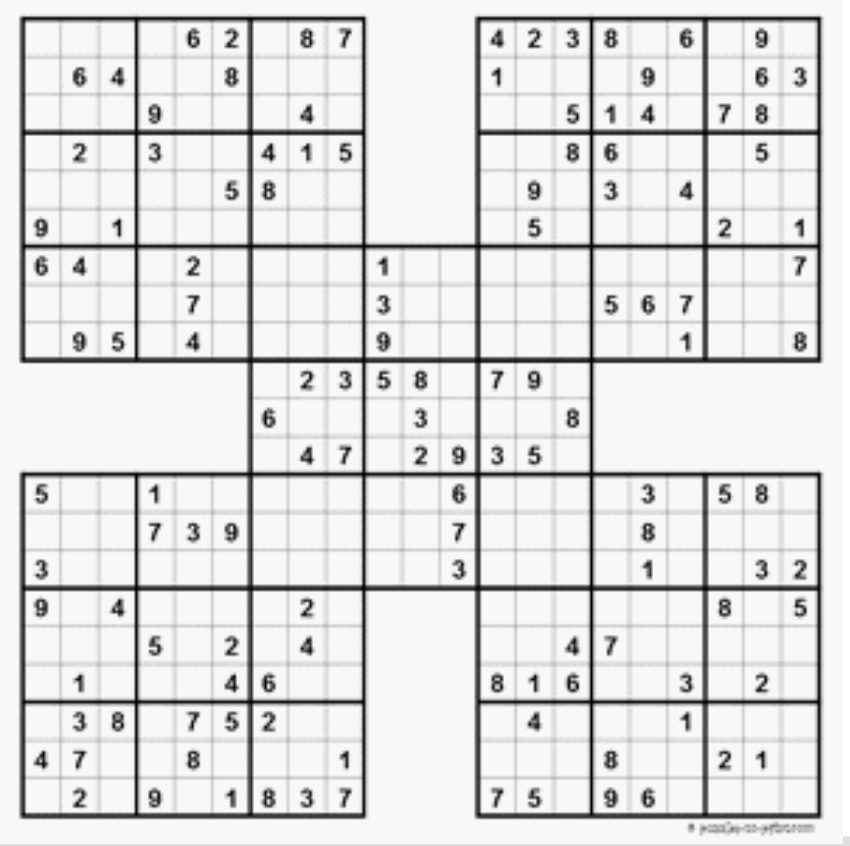
- **Example: Sudoku**
- Total number of possible assignments:
  - $2^{9\times9\times9} = 2^{729} = \mathbf{2.8 \times 10^{219}}$
  - ***How would you solve it?***

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | | | 8 | | 1 | | | 2 |
| 2 | | 1 | | 3 | | 6 | | 4 |
| | | | 2 | | 4 | | | |
| 8 | | 9 | | | | 1 | | 6 |
| | 6 | | | | | | 5 | |
| 7 | | 2 | | | | 4 | | 9 |
| | | | 5 | | 9 | | | |
| 9 | | 4 | | 8 | | 7 | | 5 |
| 6 | | | 1 | | 7 | | | 3 |

# Teaser - SMT

- SMT solvers are **magic**!
- You describe your problem (with a bit of code), the solver finds the answer

- **Example: Samurai Sudoku**
  - ***How would you solve it?***



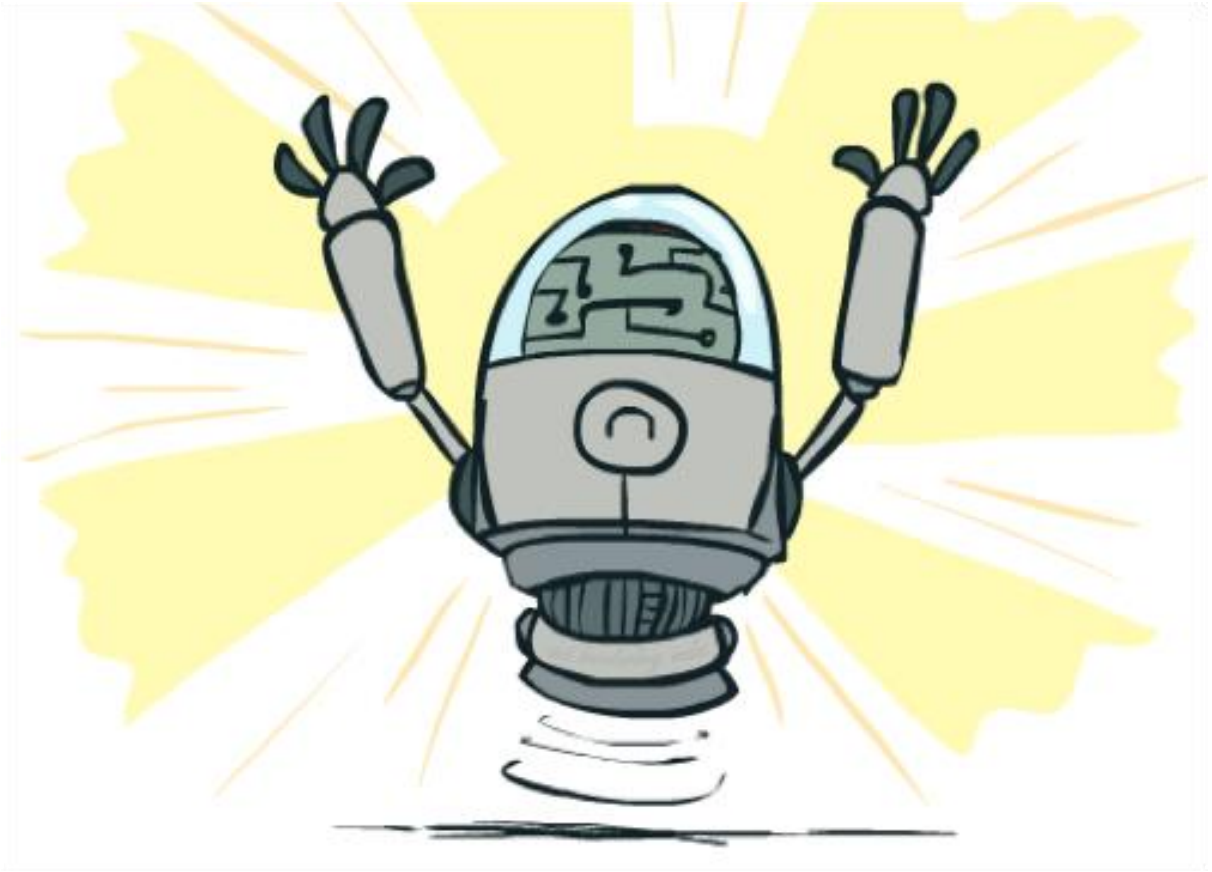
# Teaser - SMT

- SMT solvers are **magic**!
- You describe your problem (with a bit of code), the solver finds the answer

- **Example: Sudoku**
- Total number of possible assignments:
  - $2^{9\times9\times9} = 2^{729} = 2.8 \times 10^{219}$
  - **Z3 solves a Sudoku in milliseconds without the need to write an algorithm**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | | | 8 | | 1 | | | 2 |
| 2 | | 1 | | 3 | | 6 | | 4 |
| | | | 2 | | 4 | | | |
| 8 | | 9 | | | | 1 | | 6 |
| | 6 | | | | | | 5 | |
| 7 | | 2 | | | | 4 | | 9 |
| | | | 5 | | 9 | | | |
| 9 | | 4 | | 8 | | 7 | | 5 |
| 6 | | | 1 | | 7 | | | 3 |

# Thank You!
# Questions?



Discord