

PX 2018: Seminar on Programming Experience

Jens Lincke, Stefan Ramson, Patrick Rein, Marcel Taeumel, Robert Hirschfeld

Software Architecture Group
Hasso Plattner Institute
University of Potsdam, Germany

Abstract

[Active Essays](#) and [Explorable Explanations](#) is a form of interactive media that can help understand complex systems for many relevant domains. In particular, the domain of software design is filled with challenging algorithms, patterns, and systems that are hard to understand and master. In this seminar, the participants will create demos, tools, and applications based on the idea of Explorable Explanations to improve the programming experience when developing such systems. By using our live collaborative development environment, [Lively4](#), they will share, explore, and adapt the created tools and explanations --- even in unanticipated ways.

What will this Seminar be about?

- Design ideas
 - Algorithms
 - Patterns
- Found in all kinds of software systems
 - Programming languages
 - Frameworks
 - Libraries
 - Applications

How to present the idea

- Implement the idea in a new environment / language
- Generate visualizations and animations
- Goal: make the idea better understandable

Examples

- A* vs Dijkstra Path Finding Algorithms
- Explorable Explanations

Implementation Platforms



[Squeak \(Smalltalk\)](#)



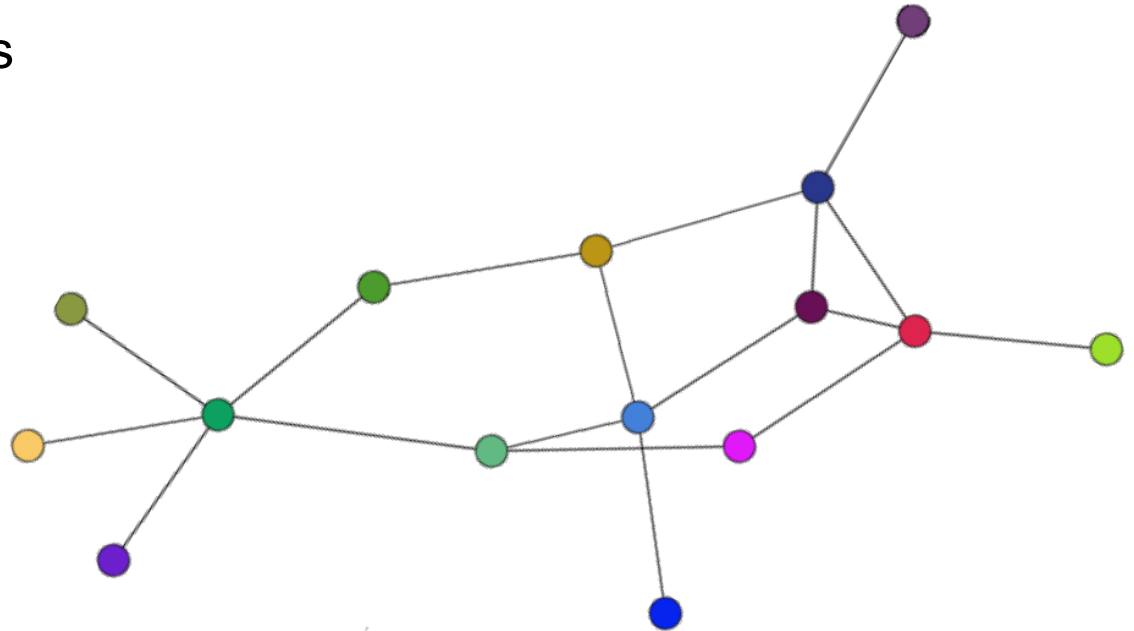
[Lively4](#): JavaScript, Web components

Topics



Topic 01: Graph Drawing

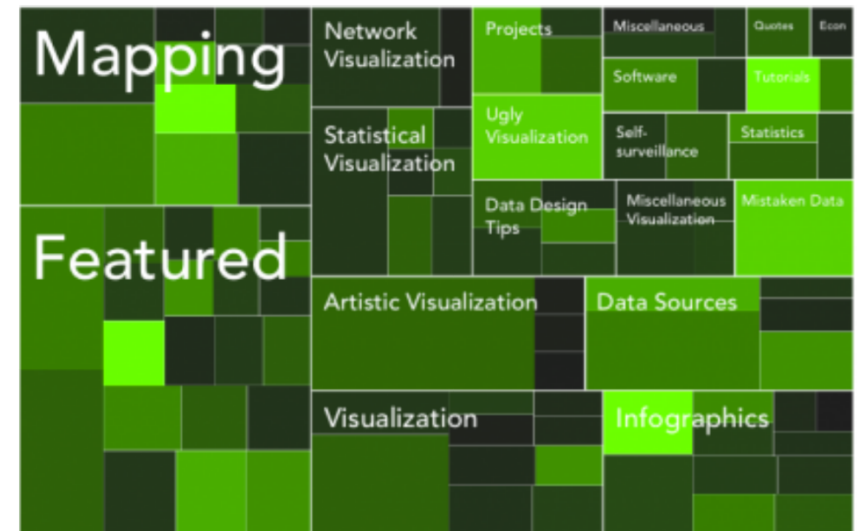
- Domain: Visualization algorithms
- Compare:
 - Simulated Annealing
 - Force Layout



- Literature
 - [wikipedia:Graph Drawing](https://en.wikipedia.org/wiki/Graph_Drawing)
 - Ron Davidson, David Harel (1996). Drawing graphs nicely using simulated annealing
 - Thomas M. J., Fruchterman Edward M. Reingold (1991). Graph drawing by force-directed placement

Topic 02: Stable Treemaps

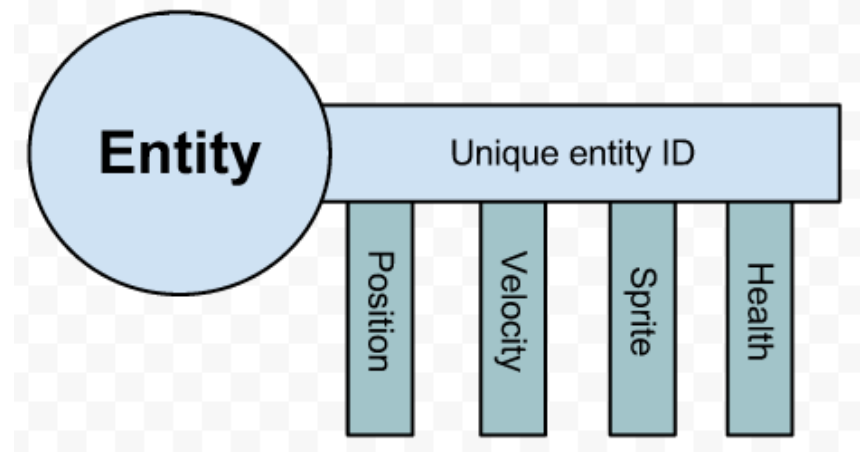
- Domain: Visualization Algorithms
- e.g. Voronoi tree maps



- Literature
 - see [Sebastian Hahn @ hpi](#)

Topic 03: Entity Component System

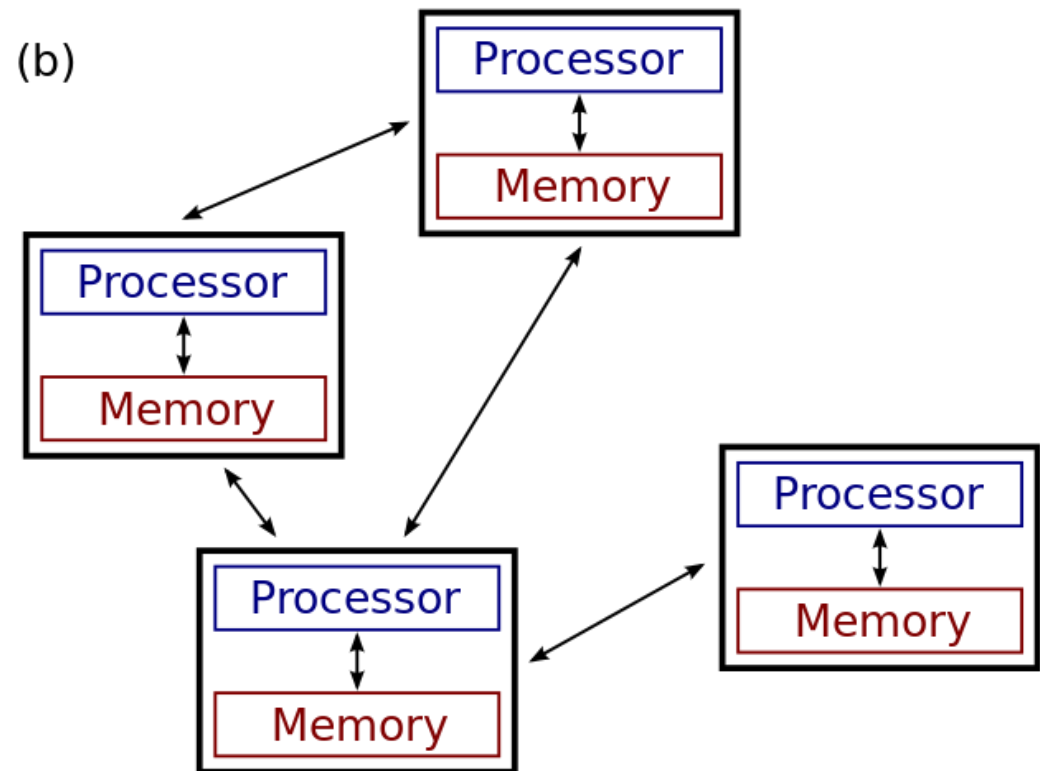
- Domain: Software Engineering
- Links/Resources



- Literature
 - [wikipedia:Entity Component System](#)
 - [Nick PrÃ¼hs \(2014\) Component-Based Entity Systems in Spielen](#)
 - [Michael House \(2012\) What is the role of "systems" in a component-based entity architecture](#)
 - [Chris Granger \(2012\). Anatomy of a Knockout](#)

Topic 04: Conflict-free replicated data type

- Domain: Software Engineering / Distributed Computing



- Literature
 - [wikipedia:Conflict-free replicated data type](https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type)

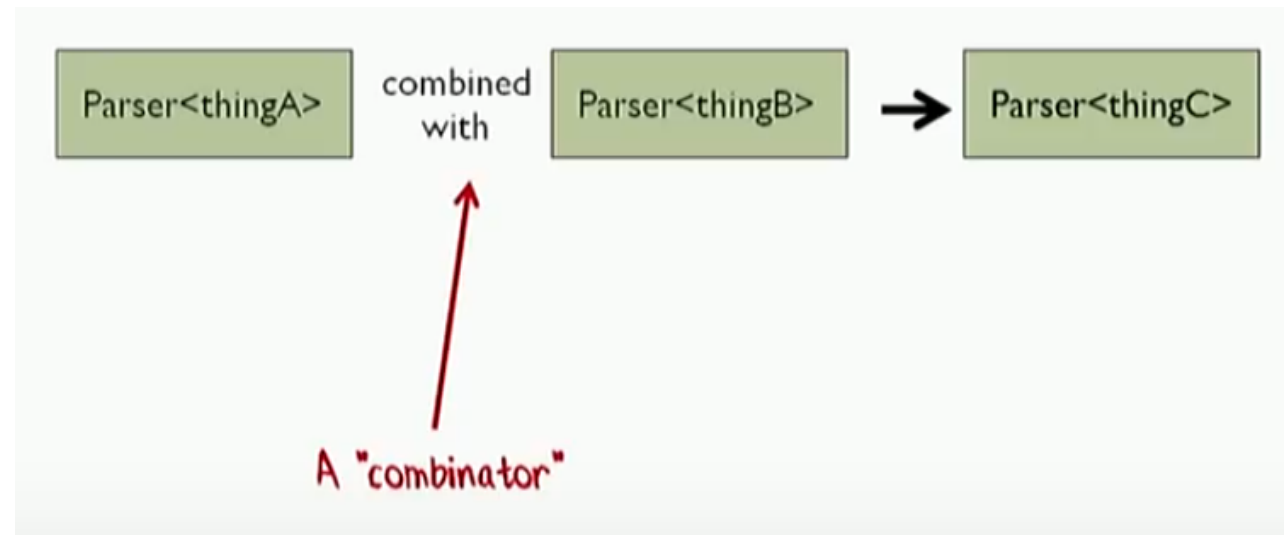
Topic 05: Regular Expressions

- Domain: Algorithms
- Implementation Strategies
 - Deterministic Finite Automatons (DFAs)
 - Nondeterministic Finite Automatons (NFAs)
 - Backtracking
- Literature
 - [wikipedia:Regular Expression](https://en.wikipedia.org/wiki/Regular_expression)



Topic 06: Parser Combinator

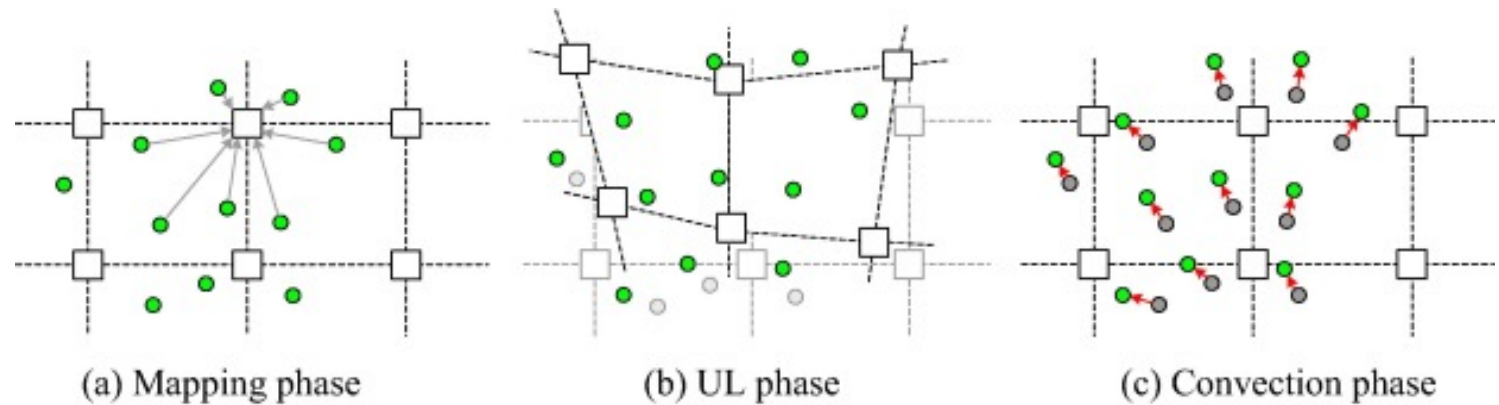
- Domain: Software Engineering



- Literature
 - [wikipedia:Parser combinator](https://en.wikipedia.org/wiki/Parser_combinator)

Topic 07: Material Point Method

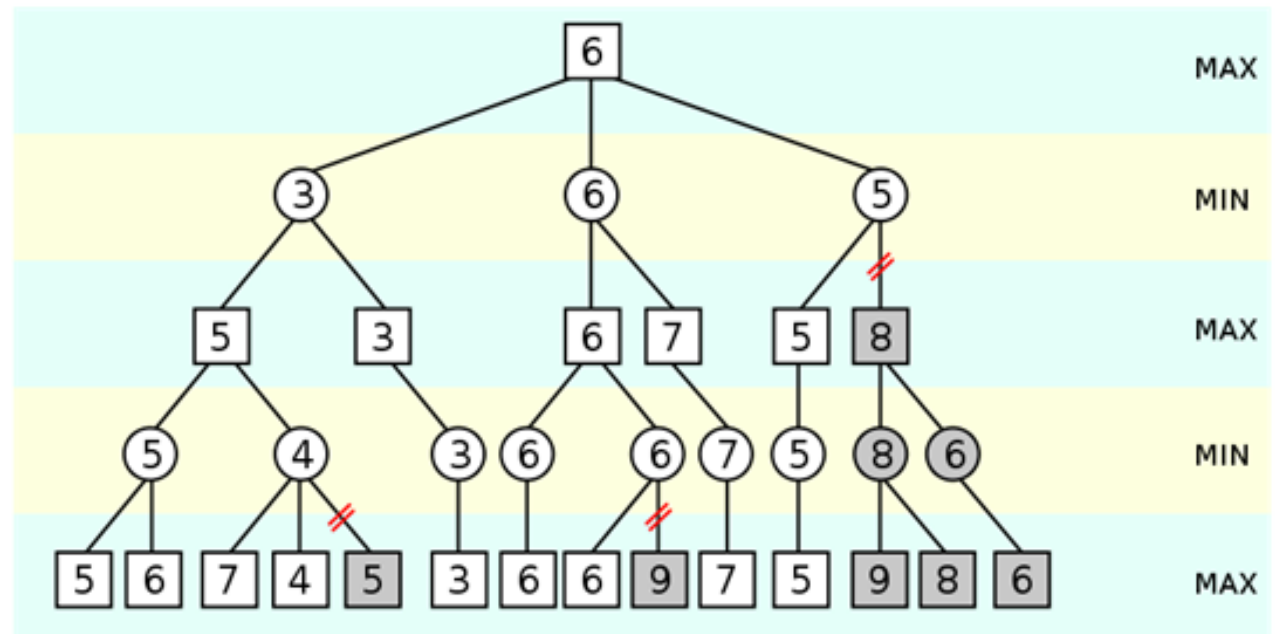
- Domain: Simulation Algorithms
- [Example](#)



- Literature
 - [wikipedia:Material Point Method](https://en.wikipedia.org/wiki/Material_Point_Method)

Topic 08: Alpha-beta Pruning

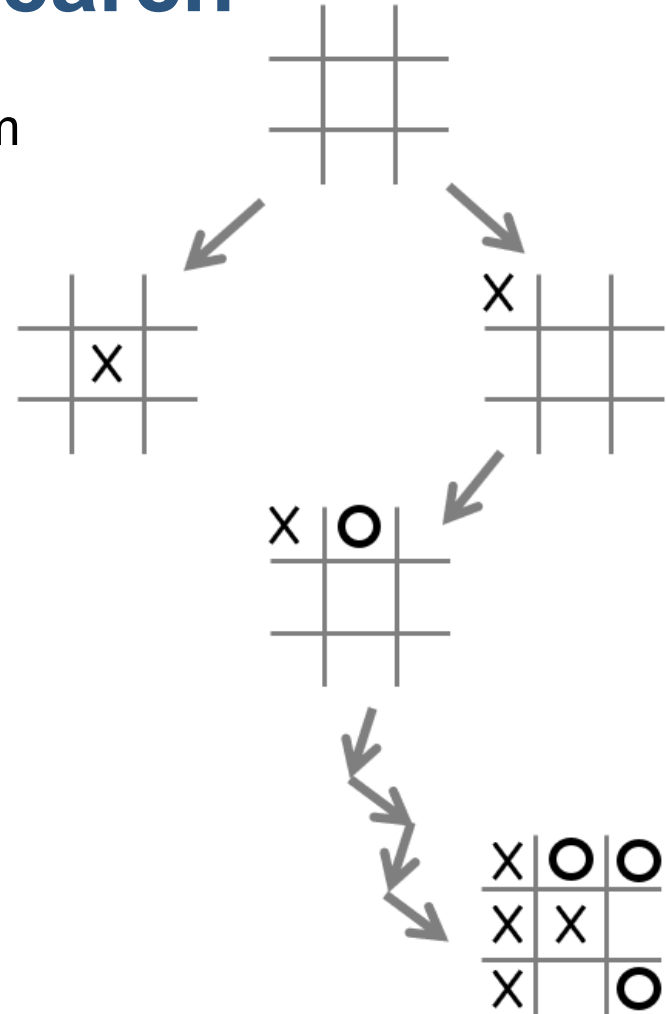
- Domain: Artificial Intelligence / Search Algorithm



- Literature
 - [wikipedia:Alpha-beta Pruning](https://en.wikipedia.org/wiki/Alpha-beta_Pruning)
 - Knuth, D. E., and Moore, R. W. (1975). "An Analysis of Alpha-Beta Pruning"
 - [Chessprogramming Wiki / Alpha-Beta](#)

Topic 09: Monte Carlo Tree Search

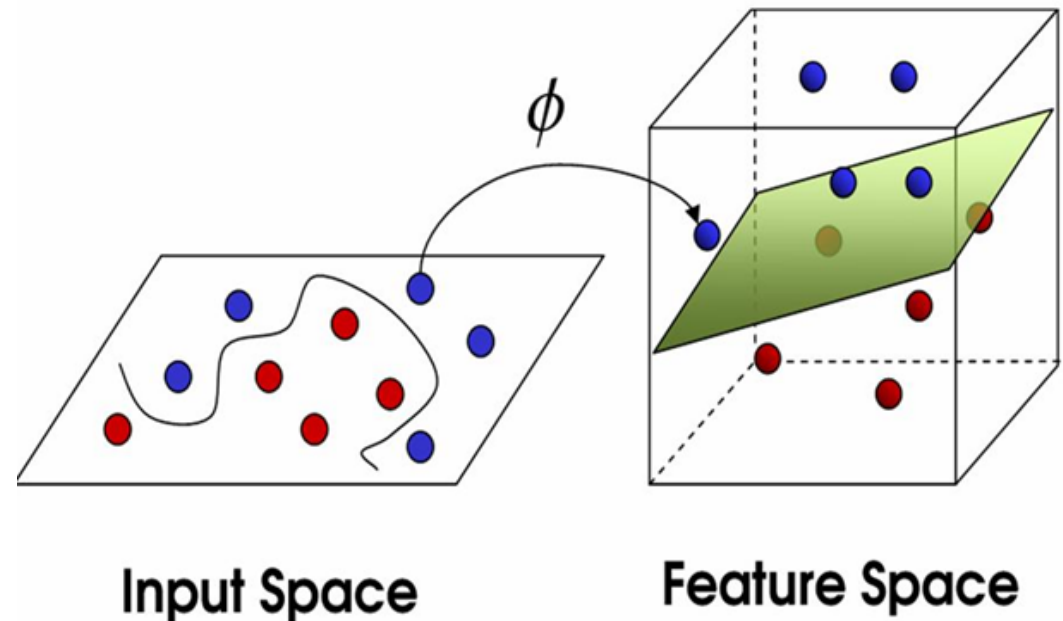
- Domain: Artificial Intelligence / Search Algorithm
- Example Application: Alpha Go KI



- Literature
 - [wikipedia:Monte Carlo Tree Search](https://en.wikipedia.org/wiki/Monte_Carlo_Tree_Search)
 - [Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search"](#)

Topic 10: Support Vector Machine

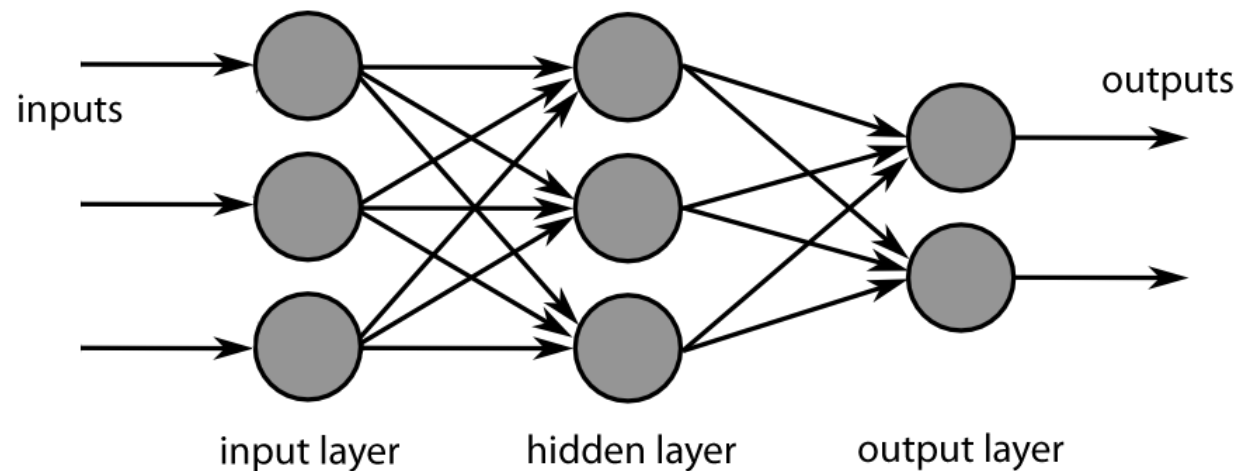
- Domain: Artificial Intelligence



- Literature
 - [wikipedia:Support Vector Machine](https://en.wikipedia.org/wiki/Support_Vector_Machine)

Topic 11: Artificial Neural Network

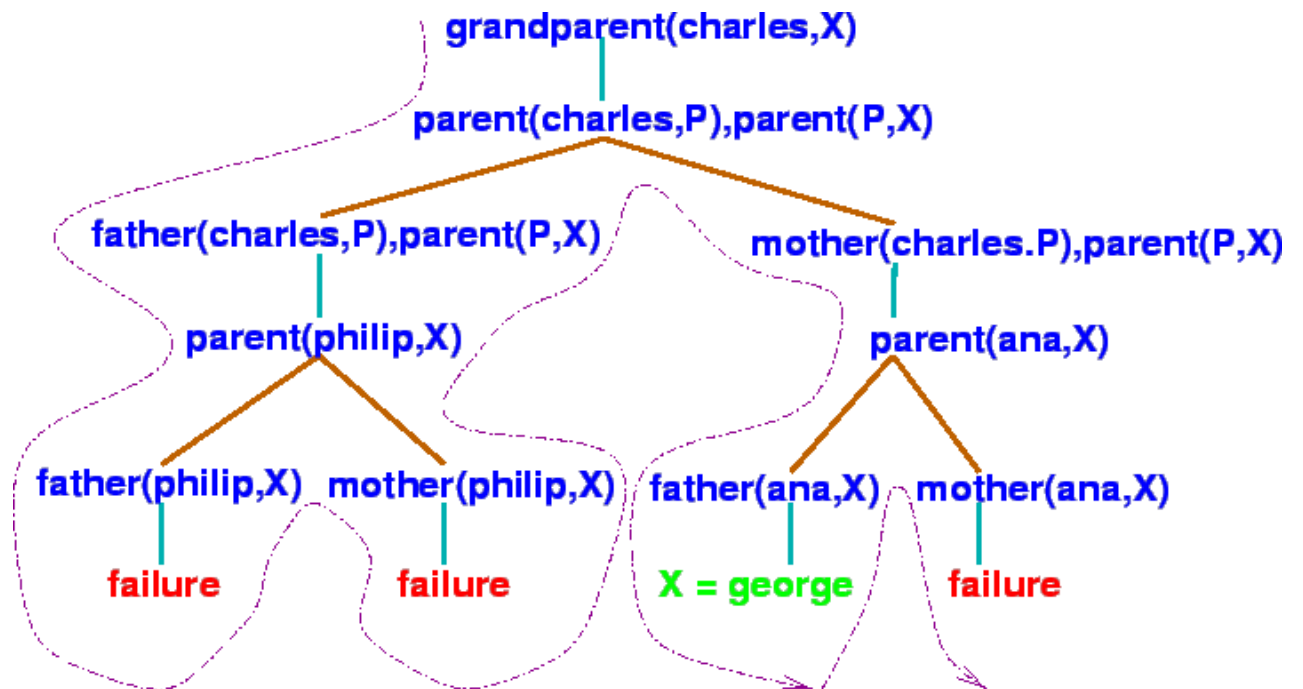
- Domain: Artificial Intelligence



- Literature
 - [wikipedia:Deep Learning](#)
 - [Michael Nielsen \(2017\). Neural Networks and Deep Learning](#)

Topic 12: Prolog / Unification

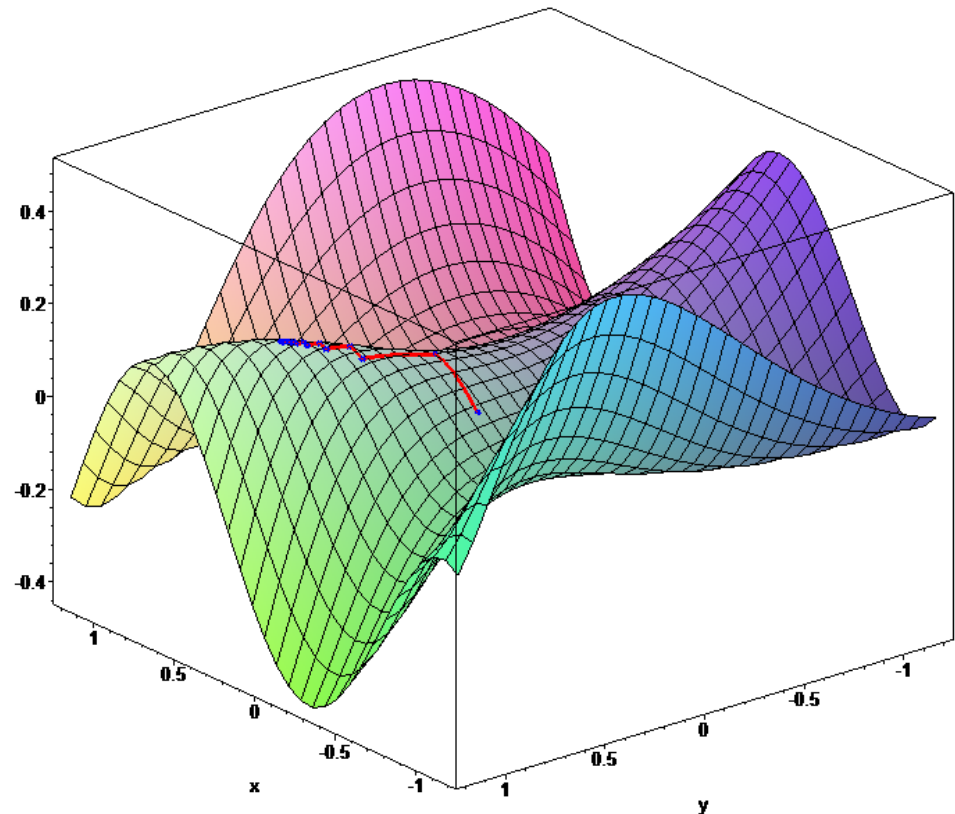
- Domain: Software Engineering / Solving / AI ...



- Literature
 - [wikipedia:Unification](https://en.wikipedia.org/wiki/Unification)
 - [wikipedia:Prolog](https://en.wikipedia.org/wiki/Prolog)

Topic 13: Constraint Solver

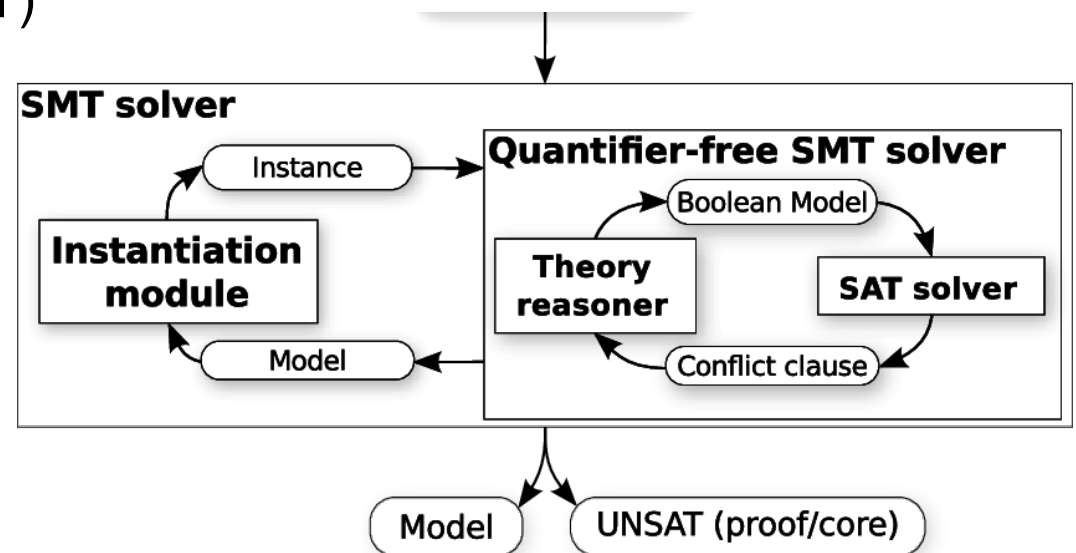
- Domain: Constraints Solving
- Compare
 - Cassowary
 - Gradient Descent
 - Back tracking



- Literature
 - [wikipedia:Gradient Descent](https://en.wikipedia.org/wiki/Gradient_Descent)
 - Greg J. Badros, Alan Borning, and Peter J. Stuckey (2001). The Cassowary Linear Arithmetic Constraint Solving Algorithm

Topic 14: Z3

- Domain: Constraints Solving
- Theorem prover
- Satisfiability modulo theories (SMT)

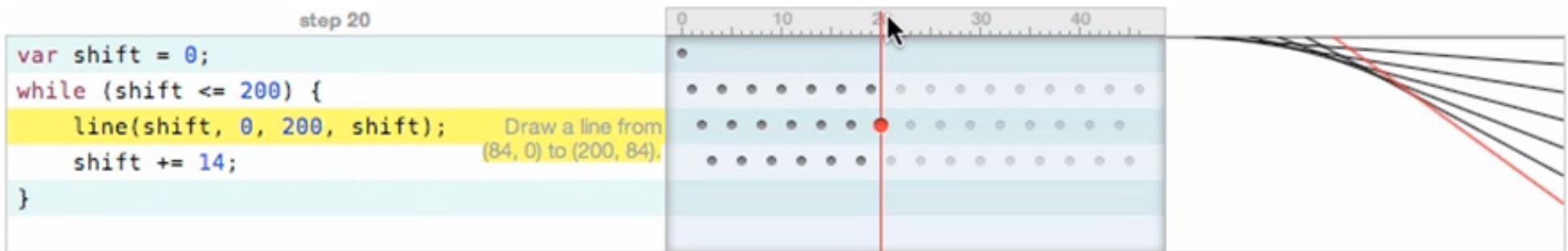


- Literature
 - Leonardo de Moura, Nikolaj Björner (2008). Z3: An efficient SMT solver
 - [github:z3](https://github.com/Z3Prover/z3)

Topic 15: Learnable Programming

1. Follow the flow

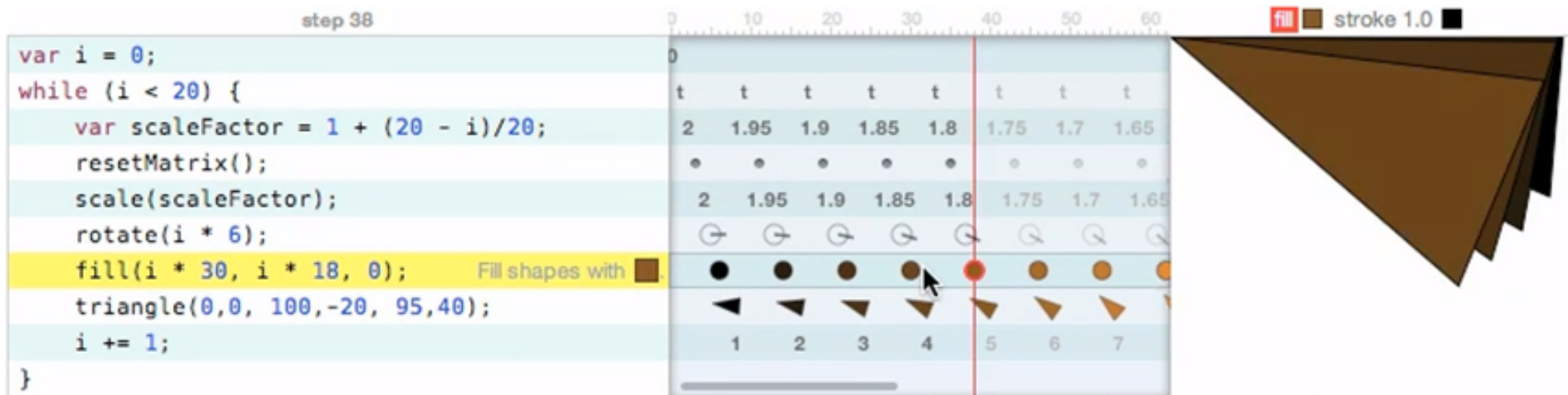
- Domain: Programming Experience
- [lively4 continuous editor prototype](#)



- Literature
 - Bret Victor 2012. Inventing on Principle.
 - Bret Victor 2012. Learnable Programming. <http://worrydream.com/>

Topic 16: Learnable Programming

2. See the state



- Literature

- Bret Victor 2012. Inventing on Principle.
- Bret Victor 2012. Learnable Programming. <http://worrydream.com/>


Topic 17: Learnable Programming

3. Create by reacting

Shapes	Color
line	background
triangle	fill
rect	stroke
ellipse	strokeWeight
bezier	

Text	Flow
text	if
textFont	for
textSize	while
	function

```
fill(0,0,0);  
rect(80,80, 40,25);  
fill(200,149,70);  
triangle(76,80, 100,20, 124,60);
```



- Literature
 - Bret Victor 2012. Inventing on Principle.
 - Bret Victor 2012. Learnable Programming. <http://worrydream.com/>

Topic 18: Learnable Programming

4. Create by abstracting

Abstract

var
function

Flow

if
for
while

```
function house (x) {  
  var y = 80;  
  rect(x, y, 40, 20);  
  triangle(x, y, 20 + x, -20 + y, 40 + x, y);  
}  
  
house(44);  
house(101);  
house(143);
```

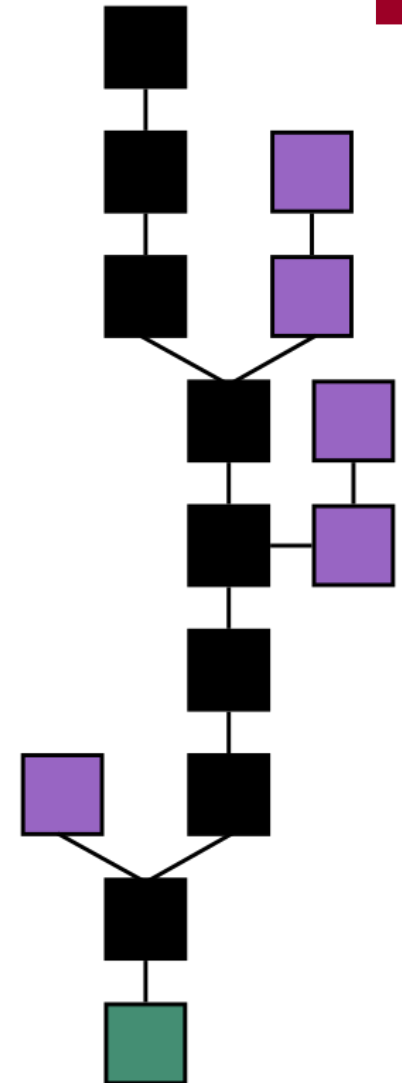


- Literature
 - Bret Victor 2012. Inventing on Principle.
 - Bret Victor 2012. Learnable Programming. <http://worrydream.com/>

Topic 19: Blockchain

- Domain: Cryptography Algorithm

- Literature
 - [wikipedia:Blockchain](https://en.wikipedia.org/wiki/Blockchain)



And ...



Propose your own topic!

Form of Presentation

- Demo / active content with text (no external slides)
- Squeak: see BookMorph etc
- Lively4: Markdown / HTML

Content of Presentation

- Present and explain problem and specific solution of the domain problem
- Use interactivity and visualizations that are supported by text
- Present your solution on a meta level: present how you visualized made it interactive

Early Feedback Presentation

- No weekly meetings, but we want to know if you are on the right track
- Solution, one very short presentation:
 - Understanding the problem / motivation?
 - Handle technical problems with both domain and presentation technology?
 - Are on the right track?
- Max 10 mins

Schedule

- Project presentation (April 11th)
- Topic assignment and Tutorial(s) (April 18th)
- Continuous consultations (in C.E)
- Early feedback presentation (10min), May 16th
- Continuous consultations (in C.E)
- Presentations (30min + discussion), 2 projects per session until July 18th

Organization

- Project-Seminar, 4 SWS, 2 students per group
- Grading
 - 6 ECTS graded credit points
 - Grade based on project and presentation
- Hand-In
 - Presentation, Screencast, Sourcecode
- Enrollment with preferred 3 topic names on or before April 16th
 - Mail to stefan.ramson@hpi.de and jens.lincke@hpi.de with PX2018 in subject
- Presentation dates determined after topics are assigned

