Intelligent assistance through Formal Logic, Management of Change, Intuitive Interfaces … and Beyond

Dr. Serge Autexier
Bremen Ambient Assisted Living Lab
Research Department
Cyber-Physical Systems (CPS)

German Research Centre for Artificial Intelligence (DFKI)
Bremen site

http://www.dfki.de/cps

Serge.autexier@dfki.de
www.dfki.de/~serge
What this talk is about

How to enable

- intelligent assistance based on rule based/formal logical reasoning directly
- through/inside familiar/intuitive user interfaces

Familiar/intuitive user interfaces
Application in Mathematics

- Support mathematician and/or mathematics students in authoring mathematical documents
- Check definitions and notation
- For proofs
  - Verify proofs
  - Complete proofs
  - Provide more details (explanations)
  - Provide hints
- Project
  - OMEGA (SFB 378, 2005 – 2007)
Transforming Editor Syntax into Proof Assistant Syntax

**Notation 10.** (Function \( \cup \)) Let \( A \) and \( B \) be sets, then we write \( A \cup B \).

**Axiom 11.** (Definition of \( \cup \))
It holds that \( \forall U, V, x. (x \in (U \cup V)) \iff (x \in U) \lor (x \in V) \).

**Definition 12.** (Function \( \cap \))
The function \( \cap : \text{set} \times \text{set} \to \text{set} \) takes two sets and returns the intersection of both sets.

**Notation 13.** (Function \( \cap \)) Let \( A \) and \( B \) be sets, then we write \( A \cap B \).

**Axiom 14.** (Definition of \( \cap \))
It holds that \( \forall U, V, x. (x \in (U \cap V)) \iff (x \in U) \land (x \in V) \).

2. Distributivity in Simple Sets

**Context 15.** We refer to the definitions and axioms of the theory SimpleSets-local.

**Theorem 16.** (Distributivity of \( \cap \))
It holds that \( \forall A, B, C. (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C)) \).

**Proof.** We begin the proof of the theorem. We have to show \( (A \cap (B \cup C)) \subseteq ((A \cap B) \cup (A \cap C)) \) and \( ((A \cap B) \cup (A \cap C)) \subseteq (A \cap (B \cup C)) \) according to the Definition of \( \subseteq \). We want to show the first subgoal. We assume \( x \in (A \cap (B \cup C)) \) in order to show \( x \in ((A \cap B) \cup (A \cap C)) \) according to the Definition of \( \subseteq \). Qed.
Combined Editor and Formal Representation

Axiom “Subset”:

\[ U, V . U \subseteq V \iff \forall x . x \in U \Rightarrow x \in V \]

“We define \( U, V . U \subseteq V \iff \forall x . x \in U \Rightarrow x \in V \)”

Axiom “Set Equality”:

\[ U, V . U = V \iff (U \subseteq V) \land (V \subseteq U) \]

Axiom “Intersection Def.”:

\[ U, V, x . (x \in U \land x \in V) \iff x \in (U \cap V) \]

“It holds \( U, V, x . (x \in U \land x \in V) \iff x \in (U \cap V) \)”

Section “Distributivity in Simple Sets”

Theorem “Distributivity of \( \cap \)”:

\[ A, B, C . A \cap (B \cup C) \iff (A \cap B) \cup (A \cap C) \]

“It holds that \( A, B, C . A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \)”

Combined representation is essential to not overwrite the users formulations when including changes from the proof assistant back into the editor.
Application in Mathematics

Axiom “Subset”:

\[ U, V \subseteq V \leftrightarrow \forall x . x \in U \Rightarrow x \in V \]

Axiom “Set Equality”:

\[ U, V . U = V \leftrightarrow (U \subseteq V) \land (V \subseteq U) \]

Axiom “Intersection Def.”:

\[ U, V, x . (x \in U \land x \in V) \leftrightarrow x \in (U \cap V) \]

Theorem “Distributivity of \( \cap \)”:

\[ A, B, C . (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C)) \]

Proof of “Distributivity of \( \cap \)”:

Subgoals

Assume \( x \in A \cap (B \cup C) \) from “Subset”.

Assume \( x \in (A \cap B) \cup (A \cap C) \) according to the Definition of \( \subseteq \). Qed.
Handling inside Theorem Prover

**Axiom “Subset”:**

\[ ! U, V . U \subseteq V \Leftrightarrow ! x . x \in U \Rightarrow x \in V \]

**Axiom “Set Equality”:**

\[ ! U, V . U = V \Leftrightarrow (U \subseteq V) \& (V \subseteq U) \]

**Axiom “Intersection Def.:”**

\[ ! U, V, x . (x \in U \& x \in V) \Leftrightarrow x \in (U \cap V) \]

**Theorem “Distributivity of \( \cap \):”**

\[ ! A, B, C . A \cap (B \cup C) \Leftrightarrow (A \cap B) \cup (A \cap C) \]

**Proof of “Distributivity of \( \cap \)”:**

**Subgoals**

- Assume \( x \in A \cap (B \cup C) \) from “Subset”.

- Assume \( x \in A \cap (B \cup C) \) by “Set Equality”:
  \[ A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C) \]

**Fact** \( x \in A \& x \in (B \cup C) \) by “Intersection Def”.

**Fact** \( x \in A \& (X \in B \mid x \in C) \) by “Union Def”.
Application in Mathematics

**Propagate Changes**

- **Push relevant changes**

---

**Axion "Subset":**
\[ U \subseteq V \iff \forall x : x \in U \implies x \in V \]

**Axion "Set Equality":**
\[ U = V \iff (U \subseteq V) \land (V \subseteq U) \]

**Axion "Intersection Def."**:
\[ U \cap V = \{ x : x \in U \land x \in V \} \]

**Section "Distributivity in Simple Sets"**

1. **Axiom "Subset"**:
\[ U \subseteq V \iff \forall x \in U : x \in V \]

2. **Axiom "Set Equality"**:
\[ U = V \iff (U \subseteq V) \land (V \subseteq U) \]

3. **Axion "Intersection Def."**:
\[ U \cap V = \{ x \in U : x \in V \} \]

**Theorem "Distributivity of \( \cap \) over \( \cup \)"**:
\[ A \subseteq (B \cup C) \land (A \cap B) \subseteq (A \cap C) \land (A \cap B) \subseteq (A \cap C) \]

Proof of "Distributivity of \( \cap \) over \( \cup \)" by "Set Equality":
- **Subproof**:
  - Proof of "Intersection Def." by "Set Equality".
  - Proof of "Union Def." by "Set Equality".

- **Fact**:
  - \( x \in A \land x \in (B \cup C) \by "Intersection Def." \)
  - \( x \in A \land x \in (B \cup C) \by "Union Def." \)

---

**Universität Bremen**

Copyright 2017, Dr. Serge Autexier, DFKI GmbH
General Principle

Surface Representation (spoken/written natural language, gestures, tablets, sensor, ...)

Model (semantic)

Model-based reasoning
General Principle: From UI to Reasoner

Surface Representation (spoken/written natural language, gestures, tablets, sensor, ...)

Model (semantic)

Model-based reasoning

π

δ
General Principle: From Reasoner to UI

Surface Representation (spoken/written natural language, gestures, tablets, sensor, ...)

Model-based reasoning

Model (semantic)

\[ \pi \]

\[ \delta \]
Services offered by Proof Assistant

Checking soundness of textbooks / articles

Interactive proof assistance inside the editor

1. Simple Sets

This theory defines the basic concepts and properties of the Theory of Simple Sets.

**Definition 1. (Type of Elements)**
First of all we define the type elem.

**Definition 2. (Type of Sets)**
Then we define the type set.

**Definition 3. (Function \( \in \))**
The function \( \in \) set \( \rightarrow \) bool takes an individual and a set and tells whether that individual belongs to this set.

**Notation 4. (Function \( \in \))** Let \( x \) be an individual and \( A \) a set, then we write \( x \in A \), \( x \) is element of \( A \), \( x \) is in \( A \) or \( A \) contains \( x \).

**Definition 5. (Function \( \subset \))**
The function \( \subset \) set \( \times \) set \( \rightarrow \) bool takes two sets and tells whether the first set is a subset of the second set.

**Notation 6. (Function \( \subset \))** Let \( A \) and \( B \) be sets, then we write \( A \subset B \).

**Axiom 7. (Definition of \( \subset \))**
It holds that \( \forall U, V, (U \subset V) \iff (\forall x \in U \Rightarrow (x \in V)) \).

**Axiom 8. (Definition of \( \subseteq \))**
It holds that \( \forall U, V, (U = V) \iff (U \subset V) \wedge (V \subset U) \).

**Definition 9. (Function \( \cup \))**
The function \( \cup \) set \( \times \) set \( \rightarrow \) set takes two sets and returns the union of both sets.

**Notation 10. (Function \( \cup \))** Let \( A \) and \( B \) be sets, then we write \( A \cup B \).

**Axiom 11. (Definition of \( \cup \))**

2. Distributivity in Simple Sets

**Context 15.** We refer to the definitions and axioms of the theory SimpleSets inside.

**Theorem 16. (Distributivity of \( \cap \))**
It holds that \( \forall A, B, C, (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C)) \).

**Proof.** We begin the proof of the theorem. We have to show \( (A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)) \) and \( ((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)) \) according to the Definition of \( \subset \). Qed.

---

Copyright 2017, Dr. Serge Autexier, DFKI GmbH
2 Classes and Sets

In Gödel-Bernays form of axiomatic set theory, which we shall follow, the primitive (undefined) notions are class, membership, and equality. Intuitively, we consider a class to be a collection \( A \) of objects (elements) such that given any object \( x \) if it is possible to determine whether or not \( x \) is a member (or element) of \( A \). We write \( x \in A \) for “\( x \) is an element of \( A \)” and \( x \not\in A \) for “\( x \) is not an element of \( A \)”.

The axiom of extensionality asserts that two classes with the same elements are equal (formally, \([x \in A \Leftrightarrow x \in B] \Rightarrow A = B \)).

A class \( A \) is defined to be a set if and only if there exists a class \( B \) such that \( A \subseteq B \). Thus a set is a particular kind of class. A class that is not a set is called a proper class. Intuitively the distinction between sets and proper classes is not too clear. Roughly speaking a set is a “small” class and a proper class is exceptionnally “large”. The axiom of class formation asserts that for any statement \( P(y) \) in the first-order predicate calculus involving a variable \( y \), there exists a class \( A \) such that \( x \in A \) if and only if \( x \) is a set and the statement \( P(x) \) is true. We denote this class \( A \) by \( \{ x \mid P(x) \} \).

A class \( A \) is a subclass of a class \( B \) (written \( A \subset B \)) provided:

\[
\text{for all } x, x \in A \Rightarrow x \in B.
\]

By the axioms of extensionality and the properties of equality:

\[
A = B \iff A \subset B \text{ and } B \subset A
\]

We first prove \( A = B \Rightarrow A \subset B \text{ and } B \subset A \): Assume (h) \( A = B \), then we have to prove (1) \( A \subset B \) and (2) \( B \subset A \): For (1), assuming \( x \in A \), we conclude \( x \in B \) from (h) and properties of equality. For (2), assuming \( x \in B \), we conclude \( x \in A \) from (h) and properties of equality. Conversely, we prove \( A \subset B \text{ and } B \subset A \Rightarrow A = B \): By Definition of \( \subset \) we know from \( A \subset B \) and \( B \subset A \) that \( x \in A \Rightarrow x \in B \) and \( x \in B \Rightarrow x \in A \) for all \( x \). Hence, \( x \in A \iff x \in B \) for all \( x \) and by extensionality follows \( A = B \). \( \square \)
Project FormalSafe (BMBF 2008 – 2010)

- Formal Methods Tools
  - Heterogeneous specification and system
  - Proof support through VSE
- Document centered development
  - Integration of formal and informal development documents
  - Dependencies and traceability
- Management of change
  - Evolutionary (agile) development
  - Formal integrity
FormalSafe Broker aka DocTIP
Related multiple documents

- Different types of documents
- Different types of reasoner syntaxes
- Automatically establish and maintain combined representations
Related multiple documents

- Different types of documents
- Different types of reasoner syntaxes
- Automatically establish and maintain combined representations
- Propagate changes from one document or reasoner to all other documents and reasoner
Parameterized Change Management

- Semantic Difference Analysis
  - Parameterized over document type specific similarity specifications
- Change Propagation
  - Representation of syntactic and semantic document parts as typed graphs
  - Parameterized over Document Specific Propagation rules (as graph rewriting rules)

Right methodology, bad scalability
FormalSafe Applications
Application: Formal Verification of C Programs

- SAMS: Formal Verification of MISRA C programs
  - annotated by preconditions, postconditions and modification information
- Used to verify algorithm computing safety zone
- Modular verification of each C-Function
- Workflow

```c
/*@ requires 0 <= w && w < sams_config.brkdist.measurements[0].v && brkconfig_OK(sams_config)
@modifies \nothing
@ensures 0 < result && result < sams_config.brkdist.length && sams_config.brkdist.measurements[result-1].v > w && w >= sams_config.brkdist.measurements[result].v
@*/
Int32 bin_search_idx_v(Float32 w);
```

```c
/*@ requires ist_SKT(t) && valid(t) && valid(i)
@assigns i[:6]
@ensures ist_SKT(i) && ((trans ^SKT_R{t}) o (trans ^SKT_R{i})) = id
@*/
void inv_trans(const SKT t, SKT i);
```
Application: Formal Verification of C Programs

- Goal
  - Avoid having to reprove everything upon each change
  - Determine proofs affected by changes in source code and/or annotations
Application: SmartTies

- Development of Safety-Critical Systems (e.g. IEC 61508) demands a variety of documents
  - Concept Paper
  - Software Failure Modes and Effects Analysis
  - Safety Requirement Specification
  - Test Plans (tables)
  - Test Suites
  - Implementation
- SmartTies tool aims to maintain these documents in a consistent way
# Fehlerbaumanalyse

## Zusammenfassung
Dieses Dokument enthält eine Fehlerbaumanalyse für das Projekt OmniProtect

<table>
<thead>
<tr>
<th>Projekt</th>
<th>OmniProtect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dokument-ID</td>
<td>DOK-HA-1</td>
</tr>
<tr>
<td>Verantwortlich</td>
<td>Christoph Lüth (cxl)</td>
</tr>
<tr>
<td>Erstellt am</td>
<td>09.01.2012</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Bearbeitungszustand</td>
<td>i.B.</td>
</tr>
<tr>
<td>Revision</td>
<td>76</td>
</tr>
<tr>
<td>Letzte Änderung</td>
<td>13.04.2012</td>
</tr>
<tr>
<td>Dokumentenablage</td>
<td>OmniProtect/fehlerbaumanalyse.docx</td>
</tr>
</tbody>
</table>
Application: Specific (BMBF 2013 – 2016)

- Specific:
- Support and maintain system development from natural language specifications down to System Level

SPECifIC Design Flow

SPEC (textbook)

NLP

Model-based reasoner 1

FSL

Formal Spec (UML)

δ

Implementation (e.g., SystemC)

class A { … }
class B { … }

ESL

Model-based reasoner m

RTL

Model (semantic)

Change Management

Code Gen.
CM Tool Architecture

Developers and Engineers

Lead Developers

Project Managers

Copyright 2017, Dr. Serge Autexier, DFKI GmbH
Tool Support for Change Management

- **Explicit semantics** approach:
  - represent syntax and semantics
  - Uniform representation by **typed graphs**
  - Graphs stored in **neo4j** graph database

- **Change impact analysis**:
  - Semantic difference analysis
  - Analysis and propagation of changes
  - Implemented **natively** in neo4j

- **Enhancement of work in FormalSafe**
  - but different implementation

**Still right methodology, now also scalable, robust**
Project SHIP (BMBF 2011 – 2013)

SHIP: Semantic Integration of Heterogeneous Processes

Research question

- How to orchestrate individual, highly specialized systems
  - Individual systems, with individual process and data models
  - Individual systems operating on different abstraction levels
  - Combination of process-oriented view with non-trivial data models
Application: Assistance Processes @ Home

- Automate assistance for specific activities of daily living
  - Cooking, reading, dressing up, …
  - Reduce energy consumption, Comfort
  - Medical assistance, Safety
- Flexible, adaptive, interruptible
- Many persons, many goals
- Many assistance processes simultaneously
- How to achieve robustness and safety?

  *Development of an ontology-based process description language (SHIP) to orchestrate and monitor environment and devices*
**SHIP (BMBF 2011 – 2013)**

**User Interfaces**

**Model (semantic)**

**Model-based reasoner**

**OWL Ontologies Modelling BAALL and Status**

\( \delta \)

**Abox updates**

**Actions as Abox updates**

**SHIP Process Language**

Dynamic Description Logic

Monitoring via LTL over OWL Queries

**Assistance Processes**

- Actions as Abox updates
- **SHIP Process Language**
- Dynamic Description Logic
- Monitoring via LTL over OWL Queries

**Propagation** to handle ramification problem via causal relationships
Application Example

- Multiple persons, multiple automatic wheelchairs
- Organisational Layer as Assistance Process
  - Handle transportation requests from individuals until completion
  - Open blocking doors, illuminate dark portions of the path
  - Avoid wheelchairs hindering each other
Application: Coordinate Rolland and AILA

Common scenario to demonstrate work from CAPIO and SHIP

*Person wearing an exoskeleton wants a scarf. Rolland and AILA are used to bring the scarf to the person. The person assists AILA in grabbing the scarf using the exoskeleton to remotely control AILA.*

SHIP Tool is used to

- receive requests
- know where the scarf is
- indicate the right shelf to AILA by blinking light
- send Rolland to the pickup position and then to the person
- control light and door during rides (as usual)
What this talk was about so far

How to enable

- intelligent assistance based on rule based/formal logical reasoning directly
- through/inside familiar/intuitive user interfaces
Current BAALL Projects
Project ModEST (1.2017 – 12.2019)

- Walker-Module for posture recognition and fall prevention
- **Problem:**
  - the correct use must be properly learned and practiced continuously
- **Goal:**
  - is to prevent latent poor postures and risk of falls.
- **Solution:**
  - distance sensors based with software to recognize poor postures
  - low-threshold feedback for posture corrections
  - in an electronic box integrated into the frame of the walker
- Funded by BMBF in program “Human-Machine-Interaction” in the scheme ”Initiatives for SMEs”

**Project Partners**

- BUDELMANN Elektronik
- GESUNDHEIT NORD KLINIKVERBUND BREMEN
- ZTOPRO
- DFKI
Learned Classifier for typical geriatric gait properties

<table>
<thead>
<tr>
<th>2 gait pattern (2gp)</th>
<th>5 gait pattern (5gp)</th>
<th>position to walker (ptw)</th>
<th>distance to walker (dtw)</th>
<th>hip flexion left (hfl)</th>
<th>hip flexion right (hfr)</th>
<th>knee flexion left (kfl)</th>
<th>knee flexion right (kfr)</th>
<th>torso flexion (tf)</th>
<th>stride symmetry (ss)</th>
<th>stride width (sw)</th>
<th>stride variability (sv)</th>
<th>stride length (sl)</th>
<th>stride height (sh)</th>
</tr>
</thead>
</table>
Project CrowdHEALTH (4.2017 - 3.2020)
Collective wisdom driving public health policies

Decision support for public health policy makers based on integrated heterogeneous health data

Funded by EU in the Call H2020-SC1-2016-CNECT „Personalised Medicine“ through Grant 727560

19 Partners

Jan Janssen
Project SMILE 4.2017 – 3.2020

… For an increasing proportion of women in Computer Science professions

„Smart Environments as a context of motivating learning opportunities for girls for a growing proportion of computer scientists by involving teachers and parents“

- For female scholars starting age of 12
- Awake and preserve interest in Computer Science
- Teach Computer Science concepts and methods

Funded by funded by the Federal Ministry for Education and Research within the support program for "Strategies for Achieving Equality of Opportunity for Women in Education and Research ("Success with MINT - New Opportunities for Women")" (FKZ 01FP1613)
Summarizing ...
Formal Logic, Management of Change, Intuitive Interfaces in …

- Project OMEGA (SFB 2005-2007)
- Project FormalSafe (BMBF 2008-2010)
- Project SHIP (BMBF 2011-2013)
- Project Specific (BMBF 2013-2016)

... and beyond

- Project ModEST (BMBF 2017-2019)
- Project CrowdHEALTH (EU 2017-2020)
- Project SMILE (BMBF 2017-2020)


Serge Autexier, Dieter Hutter, and Christoph Stahl. *An Implementation, Execution and Simulation Platform for Processes in Heterogeneous Smart Environments*, In Juan Carlos Augusto and Reiner Wichert (Ed) Fourth International Joint Conference on Ambient Intelligence, LNCS, Dublin, Ireland, Springer, December, 2013

Serge Autexier, Dieter Hutter, Christian Mandel, and Christoph Stahl. *SHIP-Tool Live: Orchestrating the Activities in the Bremen Ambient Assisted Living Lab (Demo)*, In Juan Carlos Augusto and Reiner Wichert (Ed) Fourth International Joint Conference on Ambient Intelligence, LNCS, Dublin, Ireland, Springer, December, 2013


Serge Autexier, Christoph Benzmüller, Armin Fiedler, Henri Lesourd. *Integrating Proof Assistants as Reasoning and Verification Tools into a Scientific WYSIWIG Editor*, In David Aspinall and Christoph Lüth (Ed) Proceedings of UITP’06, ENTCS, January, 2006


Thank you.