**Brief overview over (some) activities in Bernd Krieg-Brückner’s group**

<table>
<thead>
<tr>
<th>Has CASL (DFG) CASL + Haskell</th>
<th>Lutz Schröder, Christian Maeder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monads, Isabelle</td>
<td>Christoph Lüth</td>
</tr>
<tr>
<td>CASL + CSP</td>
<td>Markus Roggenbach</td>
</tr>
<tr>
<td>CSP: Java to CSP; Timed CSP, Hybrid CSP</td>
<td>Hui Shi</td>
</tr>
<tr>
<td>Multiple: heterogeneous specification (DFG)</td>
<td>Till Mossakowski, Klaus Lüttich</td>
</tr>
<tr>
<td>Abstraction and re-use of formal developments (DFG)</td>
<td>Einar Broch Johnson</td>
</tr>
<tr>
<td>UniForM: Tool integration, version control (Bremen ISP)</td>
<td>George Russell</td>
</tr>
<tr>
<td>MMISS (BMBF)</td>
<td>various people</td>
</tr>
<tr>
<td>Safe Robotics (DFG)</td>
<td>several people</td>
</tr>
</tbody>
</table>
Simple Heterogeneous Specification

Till Mossakowski

IFIP WG 1.3 meeting, January 5-9, 2002
Structure of the talk

- Introduction and motivation
- Institution morphisms and representations
- Approaches to heterogeneous specification, and their problems
- A new simplifying solution
- Internalized borrowing
- Conclusion and future work
Introduction and motivation
Motivation

- Multi-logic specifications are needed, since complex problems have different aspects that are best specified in different logics
- A combination of all the used logics would become too complex in many cases
- Different approaches can be compared
- Formal interoperability among languages and tools
A heterogeneous example

%CASL-LTL

spec SYSTEM = BUFFER and
USER
then dsort system
free types

\begin{align*}
\text{system} & \ ::= \quad \mid \quad (\text{buffer}; \ \text{user}) \\
\text{lab\_system} & \ ::= \text{START} \mid \text{OK} \mid \text{ERROR} \mid \text{tau}
\end{align*}

\forall s,s:\text{SYSTEM}
\begin{itemize}
\item \(\nabla (s, \Box \langle \lambda l. \neg l = \text{ERROR} \rangle)\)
\item \(\text{START} \quad s \rightarrow s' \Rightarrow \nabla (s', \Diamond (\langle \text{OK} \rangle \lor \langle \text{ERROR} \rangle))\)
\end{itemize}

%%% there is always a possible correct behaviour

\begin{itemize}
\item \(\text{OK} \quad (s \rightarrow s' \lor s \rightarrow s' \Rightarrow \nabla (s', \Box \langle \lambda l. \neg l = \text{OK} \lor l = \text{ERROR} \rangle))\)
\end{itemize}

%%% after starting, always the system will eventually send out either OK or ERROR

\begin{itemize}
\item \(\text{ERROR} \quad (s \rightarrow s' \lor s \rightarrow s' \Rightarrow \nabla (s', \Box \langle \lambda l. \neg l = \text{OK} \lor l = \text{ERROR} \rangle))\)
\end{itemize}

%%% OK and ERROR are sent at most once, and it cannot happen that both are sent

\ldots
%SB-CASL

System BIT
use VALUE then
{
    %CASL-LTL
dsort system
free types system ::= ___ | ___ (buffer; user)
    lab_system ::= START | OK | ERROR | tau
hide LTL-keep
dynamic func Buf_Cont : buffer; User_State :
    user_state;
proc proc_START; proc_OK; proc_ERROR; proc_tau;
    • proc_START = seq User_State := Putting_0; Buf_Cont := Empty end
    • proc_tau = if User_State = Putting_0 then
        seq User_State := Putting_1;
        Buf_Cont := Put(0, Buf_Cont') end
        elseif ...
post proc_tau : (Buf_Cont || User_State) \xrightarrow{tau} (Buf_Cont' || User_State')
%%% Specify the generated LTL relation
Then we have the following refinement

```plaintext
view v : SYSTEM
    TO { BIT hide SB-CASL with LTL-keep } end

%%% BIT is a possible run of SYSTEM
```
The CoFI representation graph

- **CASL—LTL**
  - drop
  - keep

- **LB—CASL**
  - drop
  - keep

- **SB—CASL**

- **HO—CASL**

- **FOL**

- **SubPHorn**

- **Cond**

- **HOL**

---
Institution morphisms and representations

- **Institutions** (Goguen, Burstall 1984) capture model theory
- **Institution morphisms** (Goguen, Burstall 1984) capture the intuition that one logic is **built upon** another one
- **Institution representations** (Tarlecki 1995) or **maps of institutions** (Meseguer 1987) or **institution comorphisms** (Goguen/Rosu 2001) capture the intuition that one logic is **encoded into** another one
Institutions

- category \textbf{Sign} of \textit{signatures},
- sentence functor \textbf{Sen}: \textbf{Sign} \rightarrow \textbf{Set}
- a functor \textbf{Mod}: \textbf{Sign}^{op} \rightarrow \textbf{CAT}
- satisfaction relation \(\models_{\Sigma} \subseteq |\text{Mod}(\Sigma)| \times |\text{Sen}(\Sigma)|\), such that

\[ M' \models_{\Sigma'} \text{Sen}\langle \sigma \rangle (\varphi) 
\iff 
\text{Mod}\langle \sigma \rangle (M') \models_{\Sigma} \varphi \]

or shortly

\[ M' \models_{\Sigma'} \sigma (\varphi) 
\iff 
M'\big|_{\sigma} \models_{\Sigma} \varphi \]

(satisfaction condition).
Institution morphisms

An institution morphism $\mu = (\Psi, \alpha, \beta)$ consists of

$$\begin{align*}
\text{Sign}^I & \xrightarrow{\Psi} \text{Sign}^J \\
\text{Sen}^I(\Sigma) & \xrightarrow{\alpha_{\Sigma}} \text{Sen}^J(\Psi(\Sigma)) \\
\text{Mod}^I(\Sigma) & \xleftarrow{\beta_{\Sigma}} \text{Mod}^J(\Psi(\Sigma))
\end{align*}$$

such that

$$M \models^I_\Sigma \alpha_{\Sigma}(\varphi') \iff \beta_{\Sigma}(M) \models^J_{\Phi(\Sigma)} \varphi'.$$
Institution representations

An institution representation $\mu = (\Phi, \alpha, \beta)$ consists of

$$\text{Sign}^I \xrightarrow{\Phi} \text{Sign}^J$$

$$\text{Sen}^I(\Sigma) \xrightarrow{\alpha_\Sigma} \text{Sen}^J(\Phi(\Sigma))$$

$$\text{Mod}^I(\Sigma) \xleftarrow{\beta_\Sigma} \text{Mod}^J(\Phi(\Sigma))$$

such that

$$M'|=^J_{\Phi(\Sigma)} \alpha_\Sigma(\varphi) \iff \beta_\Sigma(M')|=^I_\Sigma \varphi.$$
Approaches to heterogeneous specification, and their problems
Grothendieck institutions (Diaconescu 2000)

- A graph (category) of institutions and morphisms can be flattened into a Grothendieck institution
- Signatures are pairs (Institution, Signature in the institution)
- Signature morphisms \((\mu, \sigma): (I_1, \Sigma_1) \rightarrow (I_2, \Sigma_2)\) consist of 
  \(\mu: I_2 \rightarrow I_1\) and \(\sigma: \Sigma_1 \rightarrow \Phi_1^\mu(\Sigma_2)\)
- Sentences, models and satisfaction are taken component wise
- \(\text{Sen}(\mu, \sigma) = \text{Sen}^I_1(\Sigma_1) \xrightarrow{\text{Sen}^I_1(\sigma)} \text{Sen}^I_2(\Phi_1^\mu(\Sigma_2)) \xrightarrow{\alpha_{\Sigma_2}^\mu} \text{Sen}^I_2(\Sigma_2)\)
- \(\text{Mod}(\mu, \sigma) = \text{Mod}^I_1(\Sigma_1) \xleftarrow{\text{Mod}^I_1(\sigma)} \text{Mod}^I_2(\Phi_1^\mu(\Sigma_2)) \xleftarrow{\beta_{\Sigma_2}^\mu} \text{Mod}^I_2(\Sigma_2)\)
- Slogan: Heterogeneous specification is just specification over the Grothendieck institution
A heterogeneous specification language
(Tarlecki 1998)

- **translate** and **derive** along institution morphisms — this is just ordinary **translate** and **derive** in the Grothendieck institution
- **translate** and **derive** along institution representations — this is just ordinary **translate** and **derive** in the *representation* Grothendieck institution
- How to combine both in a coherent way?
- Free union of Grothendieck institutions?
  Hmmmm. . .
May I borrow your logic?  
(Cerioli/Meseguer 1993)

- Theorem provers can be re-used against institution representations \((\Phi, \alpha, \beta): I \rightarrow J:\)

\[
\Gamma \models_\Sigma \varphi \iff \alpha(\Gamma) \models_{\Phi(\Sigma)} \alpha(\varphi).
\]

- Premise: the representations admits model expansion
Borrowing in the heterogeneous case

- The individual logics need to be represented in some "universal" logic with sufficient expressiveness and good tool support.
- The institution morphisms need to be extended to representation maps (Tarlecki 1995, 1998).
- This can also be obtained via Grothendieck representations (Mossakowski Fossacs 2002 and last IFIP 1.3 meeting).
Representation maps (Tarlecki 1995)

A representation map \((\mu, \theta): \rho_1 \rightarrow \rho_2\)
consists of
- a morphism \(\mu = (\Psi, \alpha, \beta): I_1 \rightarrow I_2\) and
- a natural transformation \(\theta: \Phi_2 \circ \Psi \rightarrow \Phi_1\)
such that

\[
\begin{align*}
\text{Sen}^{I_1} & \xrightarrow{\alpha_1} \text{Sen}^{U \cdot \Phi_1} & \text{Mod}^{I_1} & \xleftarrow{\beta_1} \text{Mod}^{U \cdot \Phi_1} \\
\text{Sen}^{I_2 \cdot \Psi} & \xrightarrow{\alpha_2 \cdot \Psi} \text{Sen}^{U \cdot \Phi_2 \cdot \Psi} & \text{Mod}^{I_2 \circ \Psi} & \xleftarrow{\beta_2 \cdot \Psi} \text{Mod}^{U \cdot \Phi_2 \cdot \Psi}
\end{align*}
\]

commute.
Simple example

- $\mu$ forgets partiality and subsorting,
- $\rho_1$ encodes partiality and subsorting,
- $\rho_2$ is just the inclusion,
- $\theta$ is just the inclusion as well.
More complex example

- $\mu$ forgets the dynamic part (states-as-algebras),
- $\rho_1$ encodes the static part like $\rho_2$, and the dynamic part via lifting to set theory,
- $\rho_2$ encodes partiality and subsorting,
- $\theta$ is just the inclusion.
However. . .

• One has to be careful to choose compatible representations (i.e. such that there are representation maps)
• More importantly: Typically, there is no single “universal” logic.

It is crucial to use specific tools to obtain good results. Therefore, we need heterogeneous proving as well.

Cf. heterogeneous bridges (Bernot/Coudert/LeGall 1996, 1999), which, however, have no clear semantical basis.

Hmmmmmmmmmmmmmm
**First idea: Representation morphisms (new)**

Different institutions can be represented in different tool-supported logics (important for obtaining specialized tool support)

$$\begin{align*}
I_1 & \xrightarrow{\rho_1} U_1 \\
I_2 & \xrightarrow{\rho_2} U_2 \\
\mu_1 & \\
\mu_2 &
\end{align*}$$

Nice theory generalizing representation maps — but even more complex...
Further problems

- Usually, there is no “maximal” logic $\Rightarrow$ the Grothendieck signature category is not cocomplete.
- (Weak) amalgamation for the Grothendieck institution is difficult to obtain.
- However, both are needed for theorem proving with structured specifications.
A new simplifying solution
Observation

- Any useful institution morphism comes along with a left adjoint of its signature translation
- In case of such an adjointness, there is a one-one correspondence between representations and morphisms (Arrais/Fiadeiro 1995)
The Arrais/Fiadeiro construction

Given a representation with components \( \Phi: \text{Sign}^I \rightarrow \text{Sign}^J \), \( \alpha: \text{Sen}^I \rightarrow \text{Sen}^J \circ \Phi \) and \( \beta: \text{Mod}^I \leftarrow \text{Mod}^J \circ \Phi^{op} \), if \( \Phi \) has a right adjoint \( \Psi \) with counit \( \varepsilon \), then

\[
\begin{align*}
\text{Sign}^I & \xrightarrow{\Psi} \text{Sign}^J \\
\text{Sen}^I \circ \Psi & \xrightarrow{\alpha \cdot \Psi} \text{Sen}^I \circ \Phi \circ \Psi \\
\text{Mod}^I \circ \Psi & \xrightarrow{\beta \cdot \Psi^{op}} \text{Mod}^I \circ \Phi^{op} \circ \Psi^{op} \\
\end{align*}
\]

is an institution morphism.
Our solution to the problems

- Just use institution representations!
- Often, representations have a right adjoint for their signature translation, leading to a morphism
- Intuition: often, the encoding comes along with a projection
- All “useful” morphisms are of this kind!
- The adjointness anyway is needed for composable signatures in the Grothendieck institution
Heterogeneous language constructs

Let $\rho = (\Phi, \alpha, \beta): I_1 \rightarrow I_2$ be a representation in the institution graph.

If $SP_1$ is an $(I_1, \Sigma_1)$-specification, then **translate** $SP_1$ by $\rho$ is a specification with

$$\text{Sig[translate } SP_1 \text{ by } \rho \text{]} = (\Phi(\Sigma_1), I_2)$$

$$\text{Mod[translate } SP_1 \text{ by } \rho \text{]} = \{ M \in \text{Mod}^{I_2}(\Phi(\Sigma_1)) | \beta_{\Sigma_1}(M) \in \text{Mod}^{I_1}(SP_1) \}$$
Language constructs (cont’d)

Let $\rho = (\Phi, \alpha, \beta): I_1 \rightarrow I_2$ be a signature-adjoint representation in the institution graph, with $\Psi$ right adjoint to $\Phi$.

If $SP_2$ is a $(I_2, \Sigma_2)$-specification, then

**derive from** $SP_1 \text{ by } \rho$ is a specification with

$$\text{Sig[derive from } SP_1 \text{ by } \rho] = (\Psi(\Sigma_2), I_1)$$

$$\text{Mod[derive from } SP_1 \text{ by } \rho] =$$

$$\{ \beta_{\Psi(\Sigma_2)}(M|_{\varepsilon_{\Sigma_2}}) \mid \in \text{Mod}^{I_2}(SP_2) \}$$
Further heterogeneous constructs. . .

. . . are not needed!

- **derive** against representations is not useful, since it works only for signatures in the target of the signature translation.

- **translate** along morphisms is not useful, since these translations are not “surjective”.

Representation transformations

Given two institution representations $\rho_1, \rho_2: I \longrightarrow J$, a representation transformation $\tau: \rho_1 \longrightarrow \rho_2$ is just a natural transformation $\tau: \Phi_1 \longrightarrow \Phi_2$ such that

$$
\begin{array}{ccc}
\text{Sen}^I & \overset{\alpha_1}{\longrightarrow} & \text{Sen}^J \cdot \Phi_1 \\
\downarrow \alpha_2 & & \downarrow \beta_1 \\
\text{Sen}^J \cdot \tau & & \text{Mod}^J \cdot \Phi_1 \\
\downarrow \beta_2 & & \downarrow \\
\text{Sen}^J \cdot \Phi_2 & & \text{Mod}^J \cdot \tau \\
\end{array}
$$

commute.
Correspondence with representations maps

**Theorem.**
Given \( \Phi \vdash \Psi: \text{Sign}^{I_2} \longrightarrow \text{Sign}^{I_1} \), the one-one correspondence of

- representations \( \rho: I_1 \longrightarrow I_2 \) (extending \( \Phi \)) and
- morphisms \( \mu: I_2 \longrightarrow I_1 \) (extending \( \Psi \))
extends to a one-one correspondence of

- pairs \( (\rho: I_1 \longrightarrow I_2, \tau: \rho_1 \longrightarrow \rho_2 \circ \rho) \) of representations and transformations, and
- representation maps \( (\mu, \theta): \rho_2 \longrightarrow \rho_1 \).
Representation based Grothendieck institutions

- Signatures \((I, \Sigma)\) with \(\Sigma \in \text{Sign}^I\),
- signature morphisms \((\rho, \sigma): (I, \Sigma_1) \rightarrow (J, \Sigma_2)\) with \(\sigma: \Phi^\rho(\Sigma_1) \rightarrow \Sigma_2\),
  modulo \((\rho_1, \tau_\Sigma) \equiv (\rho_2, \text{id})\) for \(\tau: \rho_1 \rightarrow \rho_2\),
- sentences \(\text{Sen}(I, \Sigma) = \text{Sen}^I(\Sigma)\),
- sentence translation \(\text{Sen}(\rho, \sigma) = \text{Sen}^J(\sigma) \circ \alpha^\rho_{\Sigma_1}\),
- models analogous to sentences,
- satisfaction (and entailment) defined component wise
The congruence on Grothendieck signature morphisms

Given $\tau: \rho_1 \to \rho_2: I \to J$, we equalize

\[
\begin{array}{c}
(I, \Sigma) \\
\end{array}
\begin{array}{c}
(\rho_1, \tau) \\
\end{array}
\begin{array}{c}
(\rho_2, id) \\
\end{array}
\begin{array}{c}
(J, \Phi_1(\Sigma)) \\
\end{array}
\begin{array}{c}
(id, \tau) \\
\end{array}
\begin{array}{c}
(J, \Phi_2(\Sigma)) \\
\end{array}
\]

$\mathsf{Sen}$ and $\mathsf{Mod}$ equalize this as well!
Proof rules for the Grothendieck institution

- Just flatten structured specifications and use flat proving
  Disadvantage: structure of specification is lost
- Proof calculus for structured specifications (Borzyszkowski 1998/2003)
  Disadvantage: needs Craig interpolation for the Grothendieck institution, which often does not hold
- Proof calculus for development graphs with hiding
  (Mossakowski/Autexier/Hutter 2001)
  Needs weak amalgamation for the Grothendieck institution, which often does not hold either, but which is generally weaker than Craig interpolation. Moreover, one can try to characterize those cases where weak amalgamation holds (more easily than for Craig interpolation)
Internalized borrowing
Internalize borrowing!

- Borrowing means just translation a proof goal or refinement goal into another logic, using the heterogeneous language.
- Shifting of proof goals is sound if the representation admits model expansion.
- Allows to choose the logic for carrying out proofs in a very flexible way — typically the (a) most specific logic in which the whole heterogeneous theory for the goal can still be expressed.
The rule for borrowing

\[ M \xrightarrow{\theta} M' \quad \xrightarrow{\sigma'} N' \quad \xrightarrow{\sigma} N \]
\[ M' \xrightarrow{c \theta'} N' \]

\[ \sigma' \circ \theta = \theta' \circ \sigma \]
Example

\[
\begin{array}{c}
\text{CASL } \ni \\
\rho_{\text{CASL}}, id \\
\hline
\text{HOL } \ni \\
\rho_{\text{SB-CASL}}, id
\end{array}
\]

\[
\begin{array}{c}
S P_1 \\
(id, \tau) \\
\hline
S P_2
\end{array}
\]

The theorem link \((\rho, id)\) can be reduced to the theorem link \((id, \tau)\).

The diagram commutes because of the factoring.
A more complex example

\[
\text{CASL-LTL} \ni \text{SYSTEM} \quad \text{id} \quad \text{LTL-keep} \quad \text{BIT} \in \text{SB-CASL}
\]
**Application of the borrowing rule**

\[
\text{CASL-LTL} \ni \text{SYSTEM} \\
\text{LTL} \rightarrow \text{HOL} \\
\text{HOL} \ni \text{id} \\
\text{O} \ni \text{CASL} \\
\text{BIT} \in \text{SB-CASL} \\
\text{h SB} \\
\text{LTL-keep} \\
\text{O} \ni \text{HOL}
\]
Application of Theorem-Hide-Shift
Construction of the signature for the new node

\[ (S_B, \Phi^{S_B} \Psi^{S_B}) \]

\[ (HOL, \Phi^{S_B \rightarrow HOL} \Phi^{S_B} \Psi^{S_B}) \]

\[ (HOL, id) \]

\[ (HOL, \Phi^{LTL \rightarrow HOL} \Phi^{LTL} \Psi^{S_B}) \]

\[ (HOL, \Sigma') \]

\[ \tau_1 \]

\[ \tau_2 \]

\[ \tau'_1 \]

\[ \tau'_2 \]
Flattening out of heterogeneity

- If a proof goal is sent to a prover, the theory needs to be made homogeneous
- Take the rules for development graphs with hiding to do this
- We have to add a flattening rule that allows to translate a flattenable node (i.e. without \textit{derive}) along a representation
- For most theorem provers, ordinary \textit{translate} must be flattened out as well (but not the hierarchical structuring)
Conclusion and future work
Conclusion

- Take just representations — this is all what we need
- Morphisms are special representations
- Representation maps are special representation transformations
- Heterogeneous specification can be greatly simplified
- Borrowing can be internalized ⇒ simpler and more flexible
- Development graphs provide the calculus for doing heterogeneous proofs
Future work

• (Weak) amalgamation: holdy only sometimes — when? (look at examples)
• Include programming languages (and semi-representations)
• Behavioural refinement
• Tool support: Maya
Development graph manager Maya

• Developed in Saarbrücken — development graphs have been used for industrial-scale applications (hundreds of specifications)
• Management of proof obligations
• Management of change — specifications have errors!
• Generic for arbitrary logics and provers — only the axioms for logics (Meseguer 1987) are used
• Currently being extended with hiding
• Future work: Extension of the development graph manager Maya to the heterogeneous case
• Heterogeneous management of change — this needs something like institution with symbols, since we have to track the change of symbols