Symphony und das Hierarchische Reflexion Modell

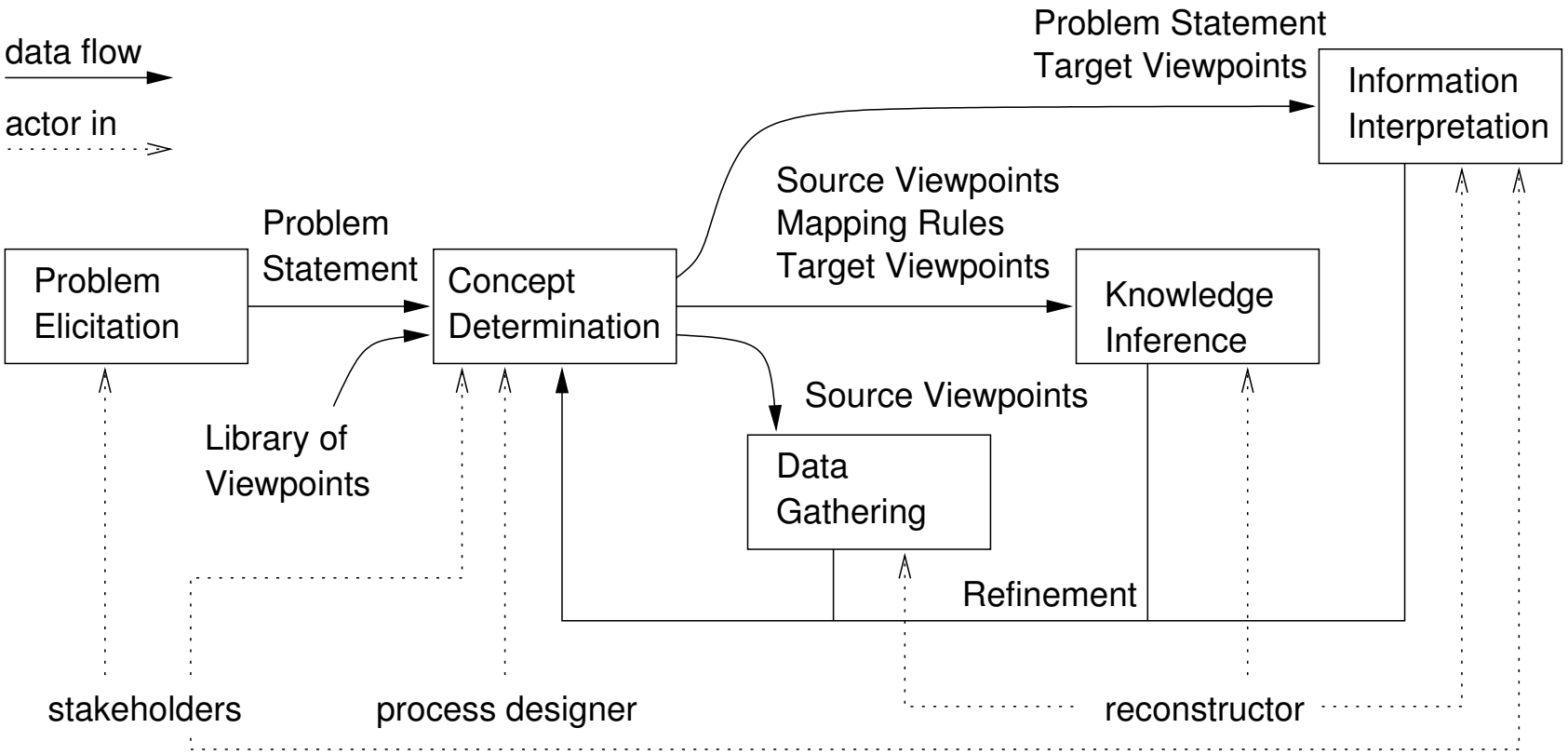
Rainer Koschke und **Daniel Simon**

Universität Stuttgart
Institut für Softwaretechnologie
Abt. Programmiersprachen und Compilerbau

WSR 2004

BAUHAUS
SOFTWARE
TECHNOLOGIES

Symphony Designebene



Problem Elicitation — Hierarchisches Reflexion Modell

Ziele:

- Architekturvalidierung von Compilern
- hypothesengetriebene Architekturrekonstruktion

Problem Elicitation — Hierarchisches Reflexion Modell

Ziele:

- Architekturvalidierung von Compilern
- hypothesengetriebene Architekturrekonstruktion
- Erweiterung Reflexion Modell von
  Murphy, Notkin, Sullivan, 1995

Concept Determination

- Useful Viewpoints:
  - Modulsicht
  - konzeptuelle Sicht
  - Reflexion Modell Sicht

Concept Determination

- Useful Viewpoints:
  - Modulsicht
  - konzeptuelle Sicht
  - Reflexion Modell Sicht
- Target Viewpoint:
  - Kombination aus Modul und Reflexion Sicht
  - später: Modul und hierarchische Reflexion Sicht

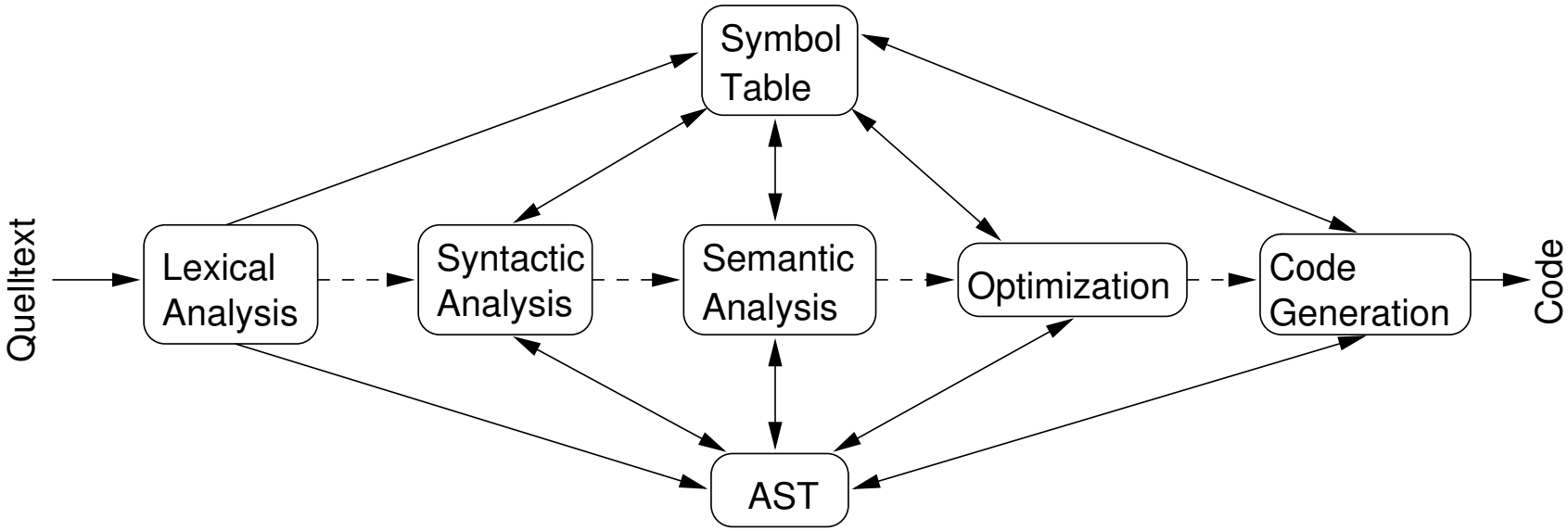
Concept Determination

- Useful Viewpoints:
  - Modulsicht
  - konzeptuelle Sicht
  - Reflexion Modell Sicht
- Target Viewpoint:
  - Kombination aus Modul und Reflexion Sicht
  - später: Modul und hierarchische Reflexion Sicht
- Mapping Rules

Concept Determination

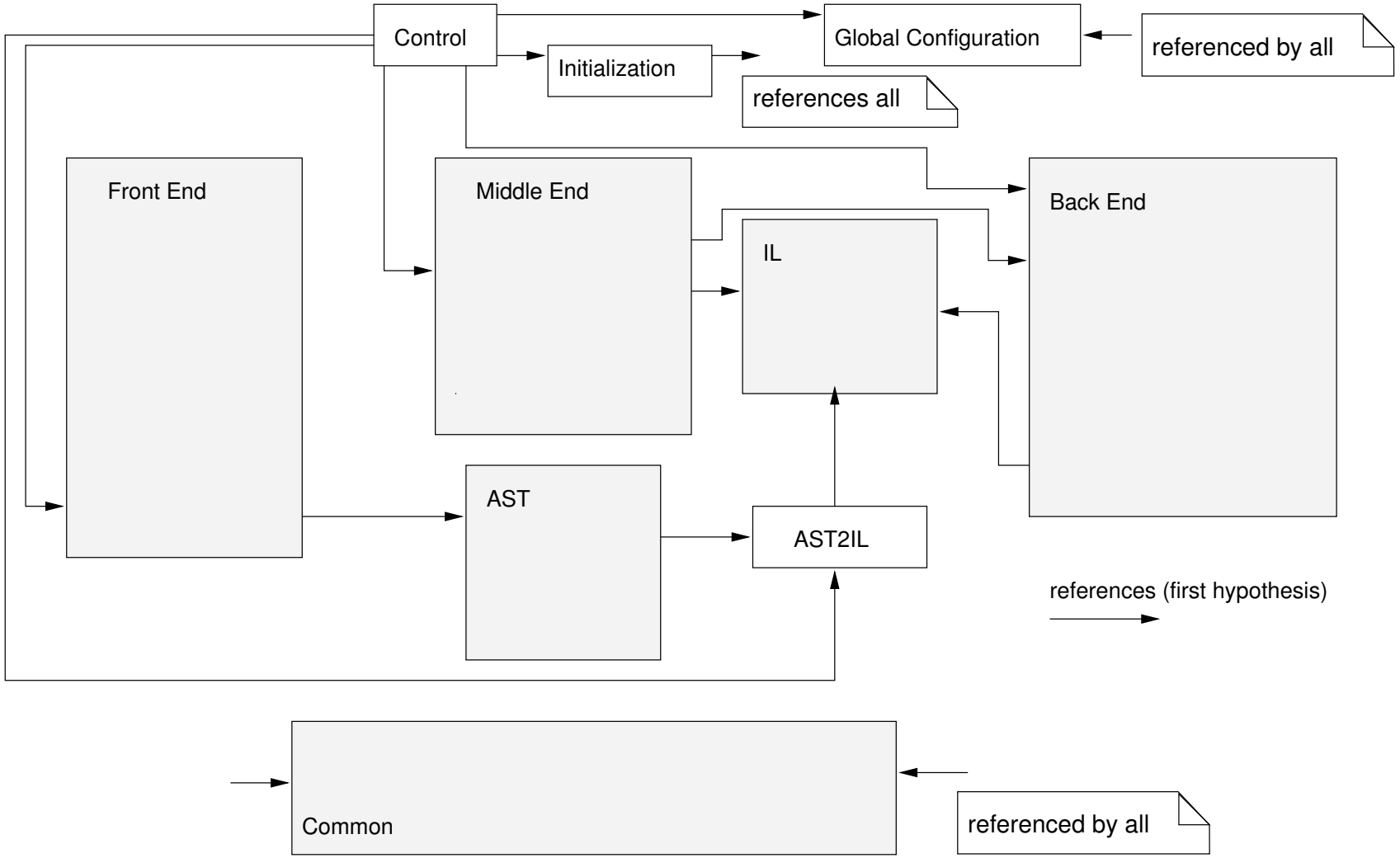
- Useful Viewpoints:
  - Modulsicht
  - konzeptuelle Sicht
  - Reflexion Modell Sicht
- Target Viewpoint:
  - Kombination aus Modul und Reflexion Sicht
  - später: Modul und hierarchische Reflexion Sicht
- Mapping Rules
- Hypothetical Views: high-level Modulsicht

Compiler — Logisch



Lehrbücher, etwa Wilhelm/Maurer, Übersetzerbau

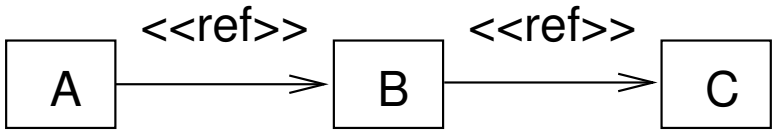
Compiler — Grobstruktur



Shaw und Garlan, 1993

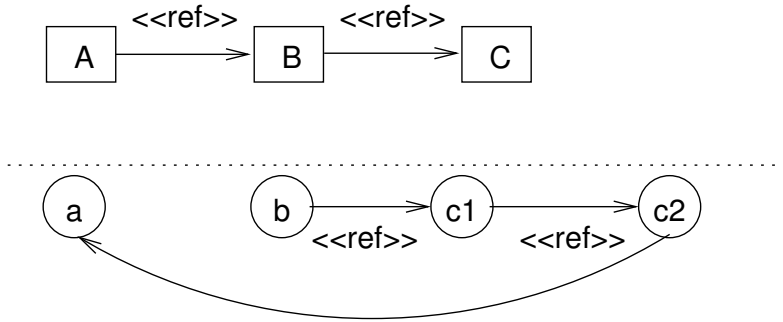
Anwendung des Reflexion Modells

1. **hypothetisches Architekturmodell erstellen**
2. Quellmodell extrahieren
3. Quellentität auf konzeptuelle Entitäten abbilden
4. Reflexion Modell berechnen
5. Verfeinern/korrigieren
   - hypothetisches Modell
   - Abbildung
   - Quellmodell



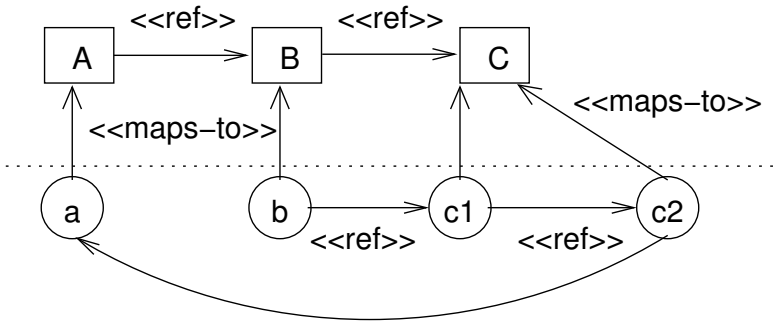
Anwendung des Reflexion Modells

1. hypothetisches Architekturmodell erstellen
2. **Quellmodell extrahieren**
3. Quellentität auf konzeptuelle Entitäten abbilden
4. Reflexion Modell berechnen
5. Verfeinern/korrigieren
   - hypothetisches Modell
   - Abbildung
   - Quellmodell



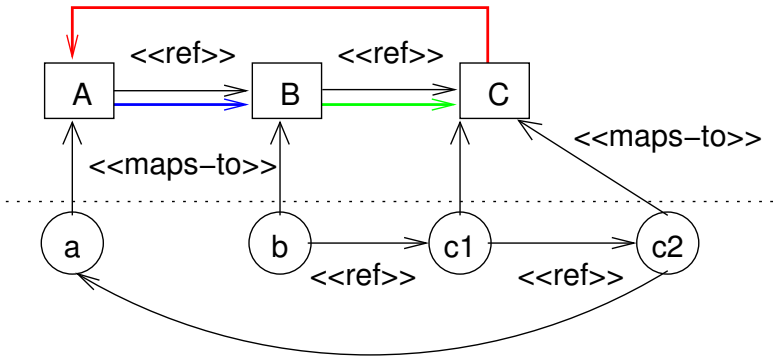
Anwendung des Reflexion Modells

1. hypothetisches Architekturmodell erstellen
2. Quellmodell extrahieren
3. **Quellentität auf konzeptuelle Entitäten abbilden**
4. Reflexion Modell berechnen
5. Verfeinern/korrigieren
   - hypothetisches Modell
   - Abbildung
   - Quellmodell



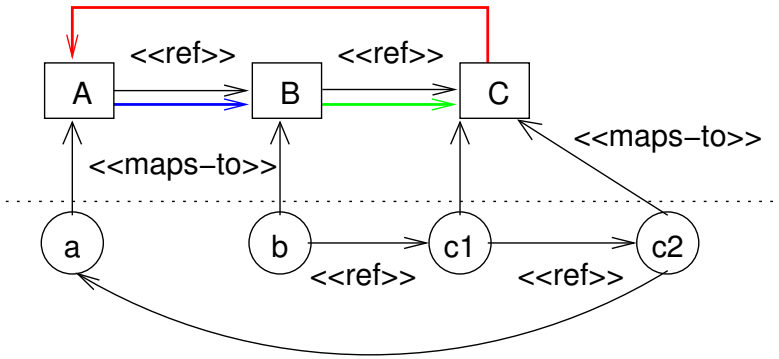
Anwendung des Reflexion Modells

1. hypothetisches Architekturmodell erstellen
2. Quellmodell extrahieren
3. Quellentität auf konzeptuelle Entitäten abbilden
4. **Reflexion Modell berechnen**
5. Verfeinern/korrigieren
   - hypothetisches Modell
   - Abbildung
   - Quellmodell



Anwendung des Reflexion Modells

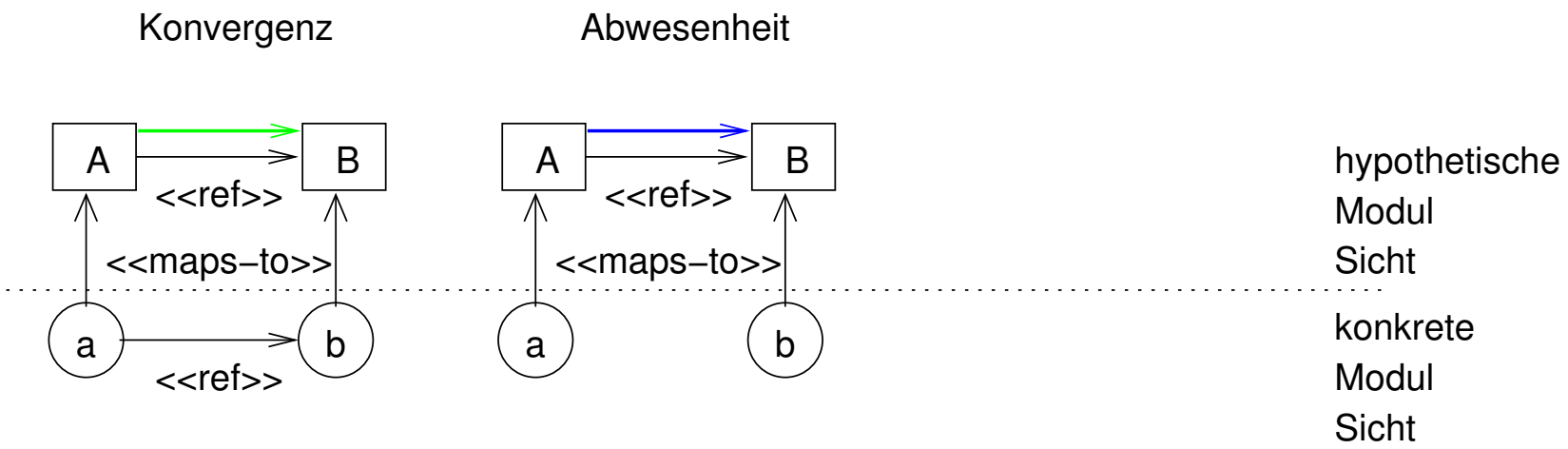
1. hypothetisches Architekturmodell erstellen
2. Quellmodell extrahieren
3. Quellentität auf konzeptuelle Entitäten abbilden
4. Reflexion Modell berechnen
5. **Verfeinern/korrigieren**
   - **hypothetisches Modell**
   - **Abbildung**
   - **Quellmodell**



Reflexion

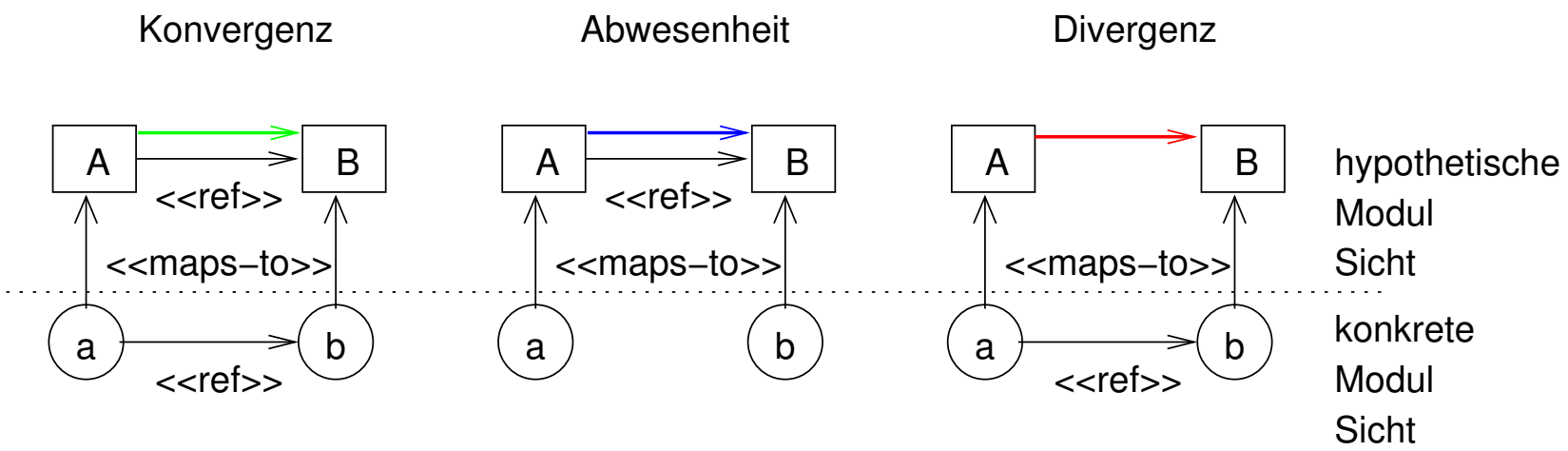
$$\begin{aligned} \textit{propagated-ref}(A,B) \quad &\Leftrightarrow \quad \exists(a,b \in M) : (\textit{ref}(a,b) \\ &\quad \wedge \textit{maps-to}(a) = A \\ &\quad \wedge \textit{maps-to}(b) = B) \\ \textit{convergence}(A,B) \quad &\Leftrightarrow \quad \textit{ref}(A,B) \wedge \textit{propagated-ref}(A,B) \end{aligned}$$

Reflexion



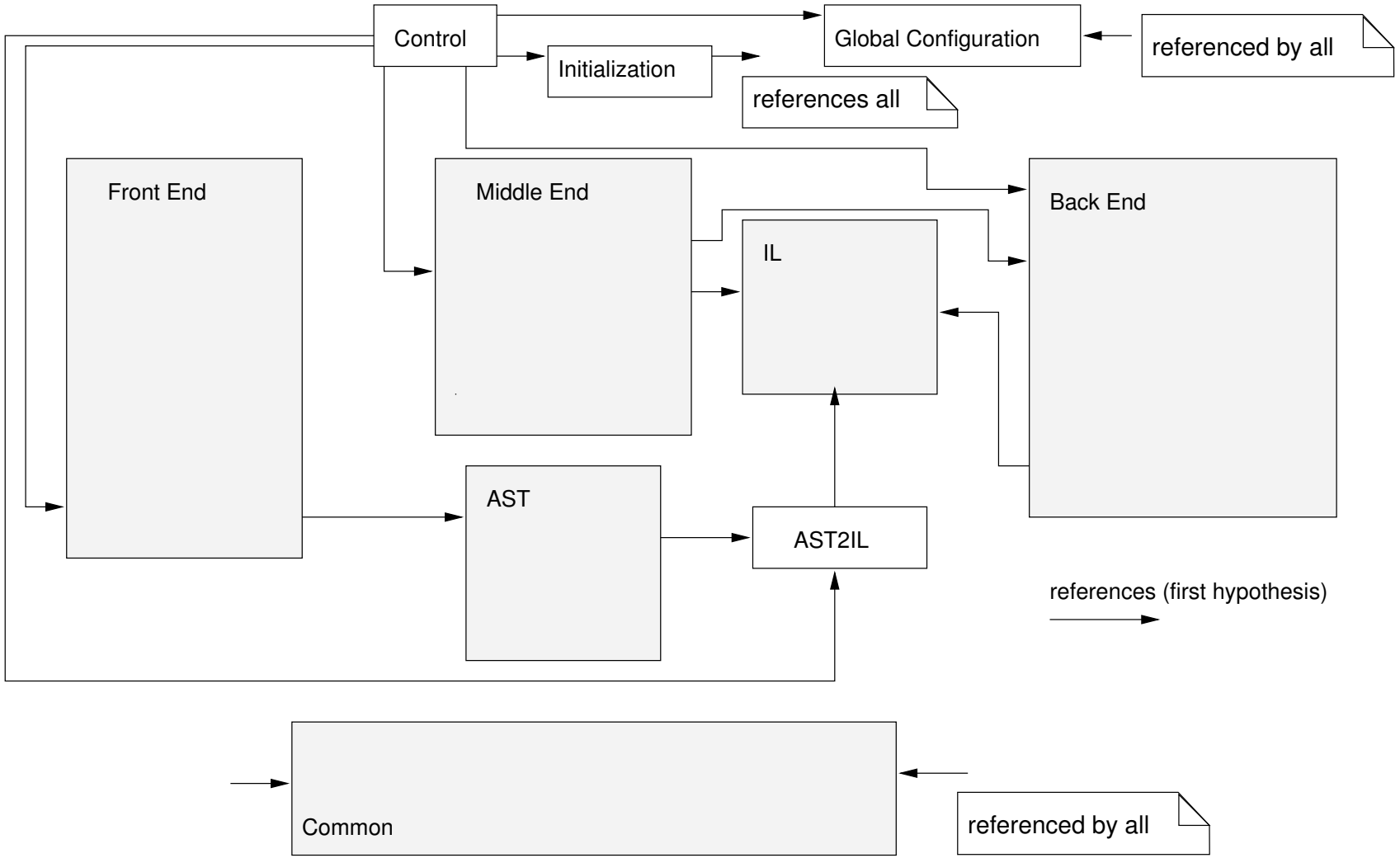
$$absence(A, B) \quad \Leftrightarrow \quad ref(A, B) \wedge \neg propagated\text{-}ref(A, B)$$

Reflexion



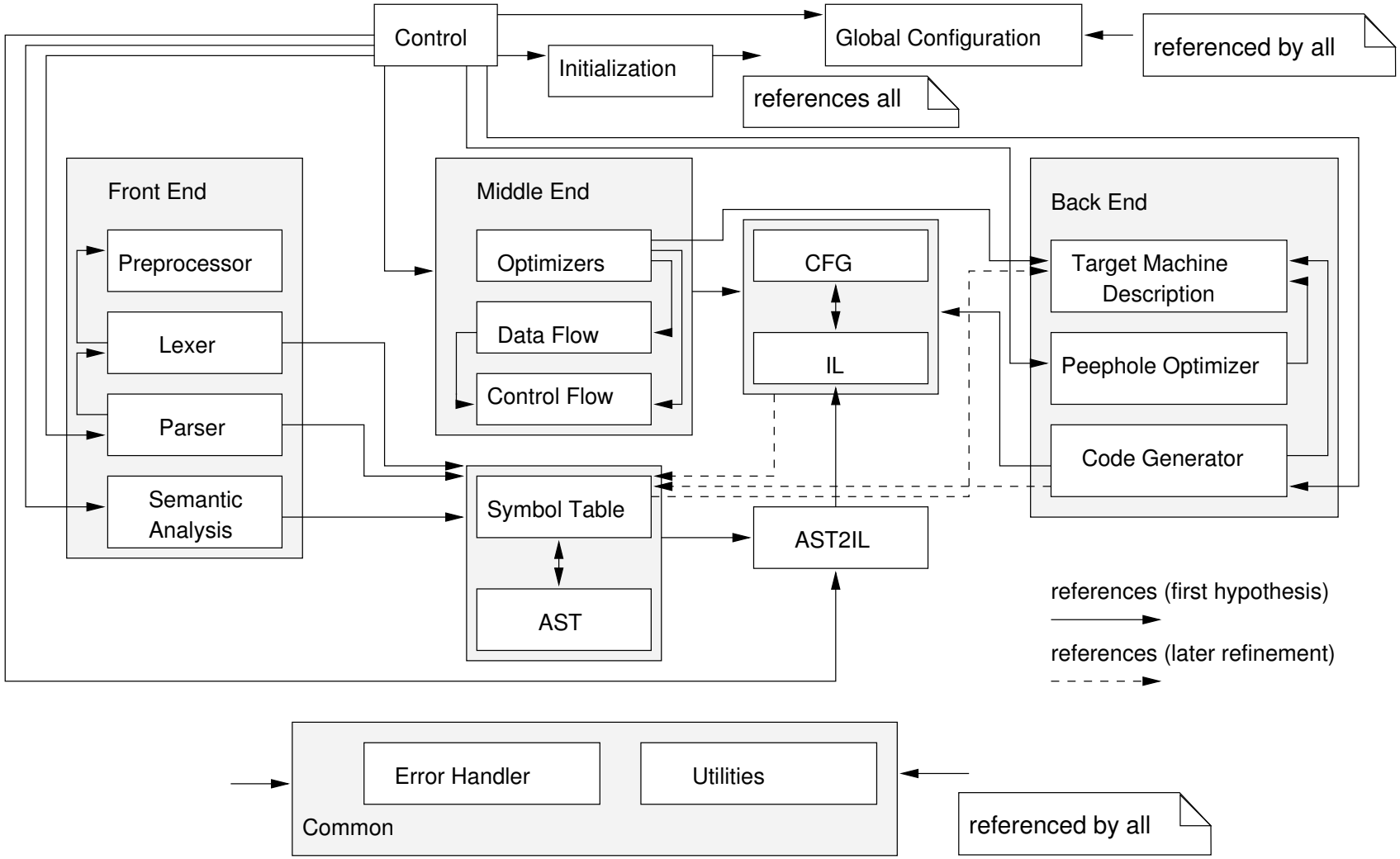
$$\mathit{divergence}(A, B) \quad \Leftrightarrow \quad \neg \mathit{ref}(A, B) \wedge \mathit{propagated\text{-}ref}(A, B)$$

Compiler — Grobstruktur



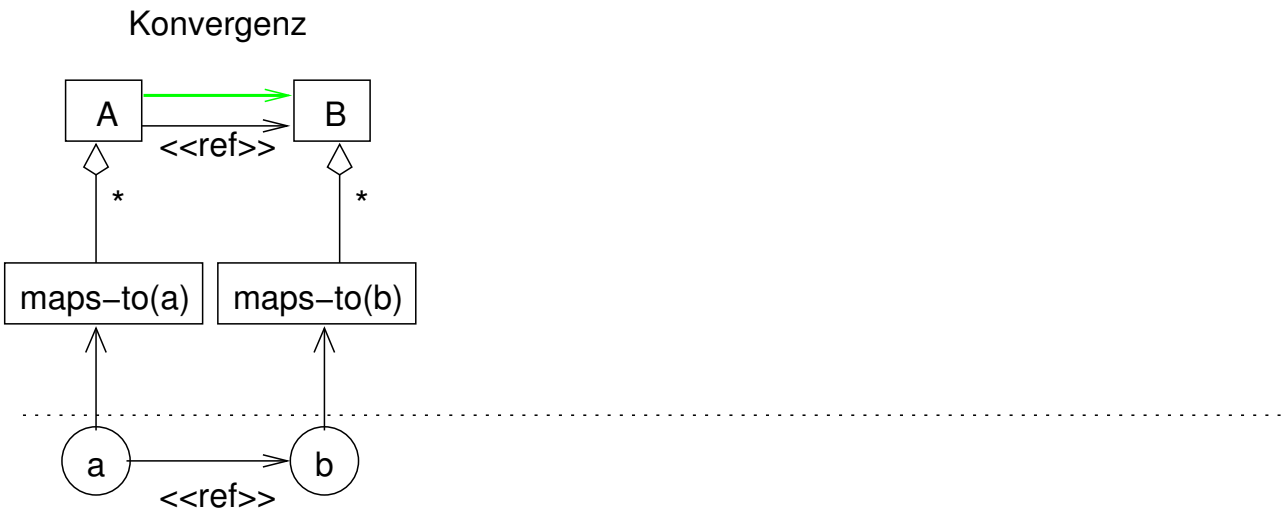
Shaw und Garlan, 1993

Compiler — Feinstruktur



Ursprüngliches Reflexion Modell hat keine Hierarchien.

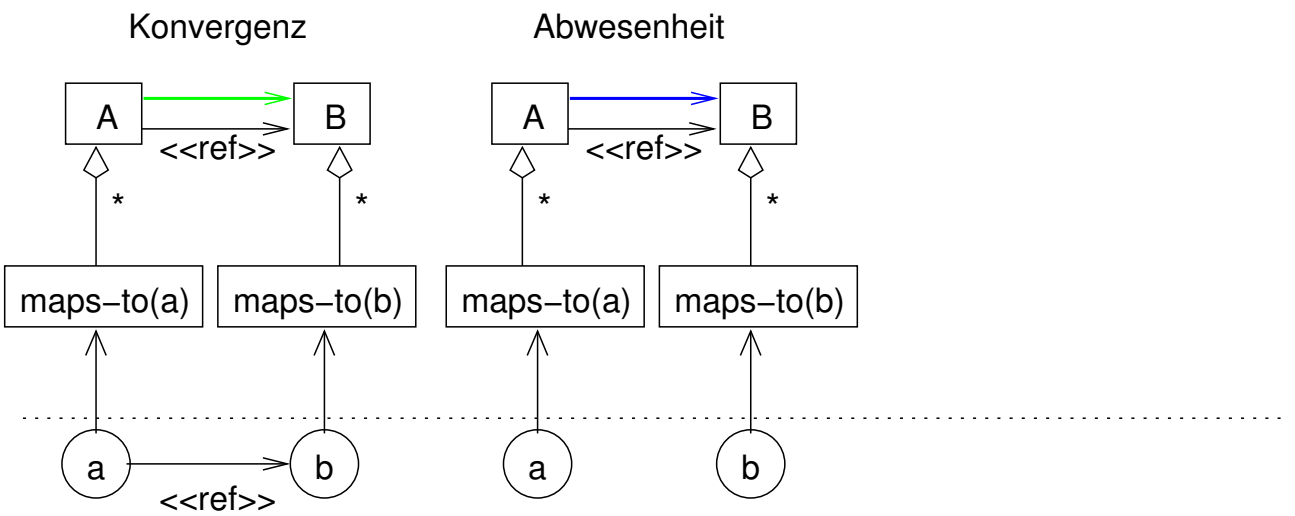
Hierarchisches Reflexion Modell



$$propagated\text{-}ref^{\uparrow}(A,B) \Leftrightarrow \exists(a,b \in M) : (ref(a,b) \wedge partof^{*}(maps\text{-}to(a),A) \wedge partof^{*}(maps\text{-}to(b),B))$$

$$convergence(A,B) \Leftrightarrow ref(A,B) \wedge propagated\text{-}ref^{\uparrow}(A,B)$$

Hierarchisches Reflexion Modell

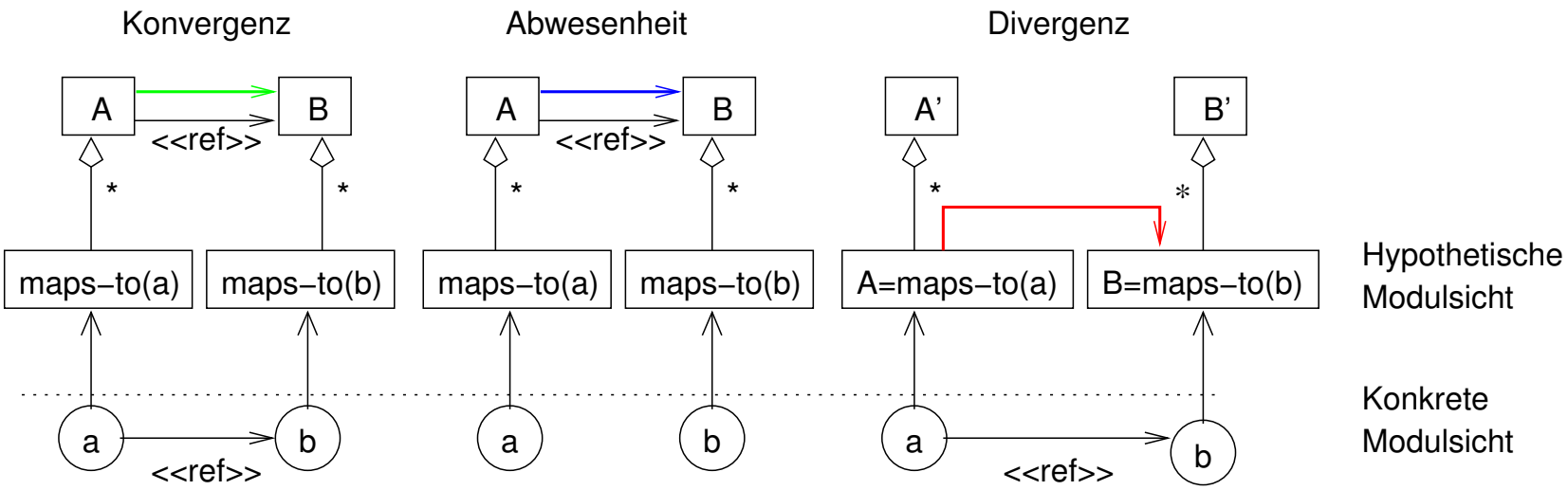


Hypothetische Modulsicht

Konkrete Modulsicht

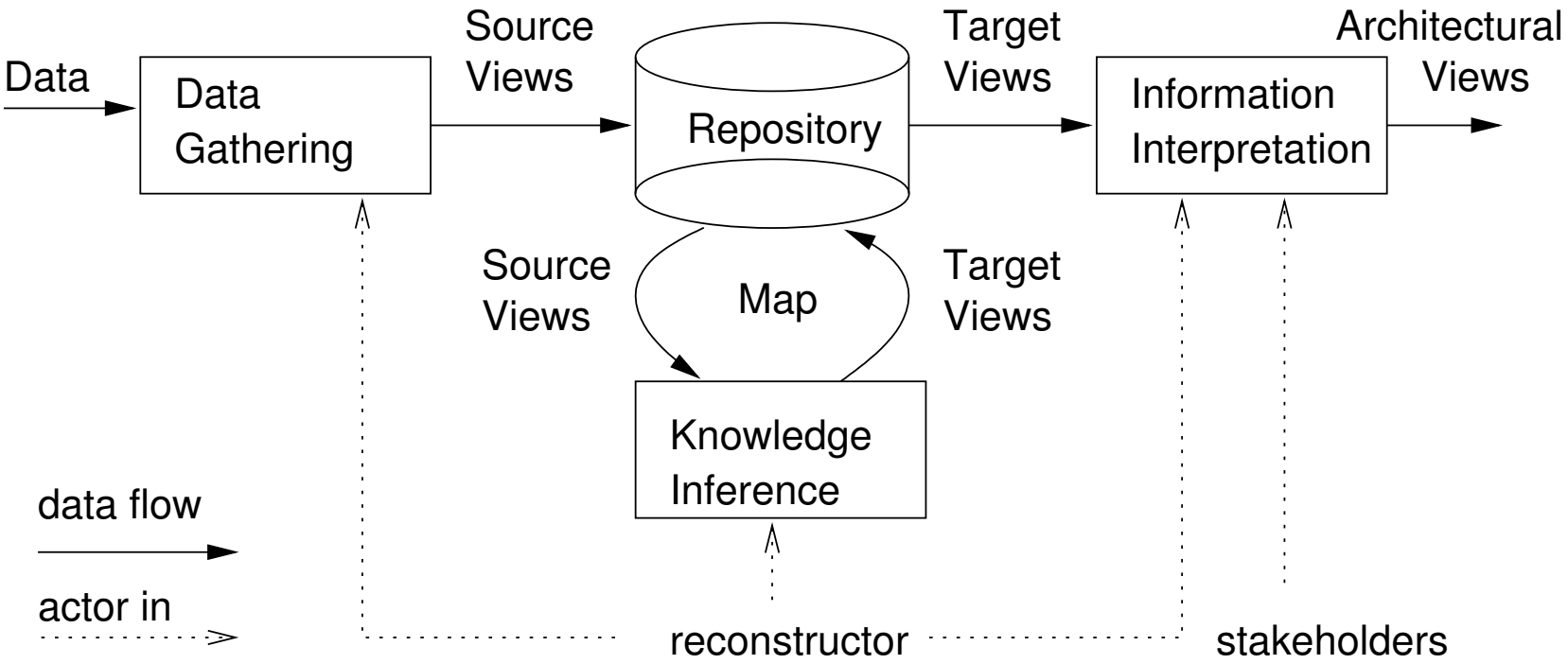
$$absence(A, B) \quad \Leftrightarrow \quad ref(A, B) \wedge \neg propagated\text{-}ref^{\uparrow}(A, B)$$

Hierarchisches Reflexion Modell

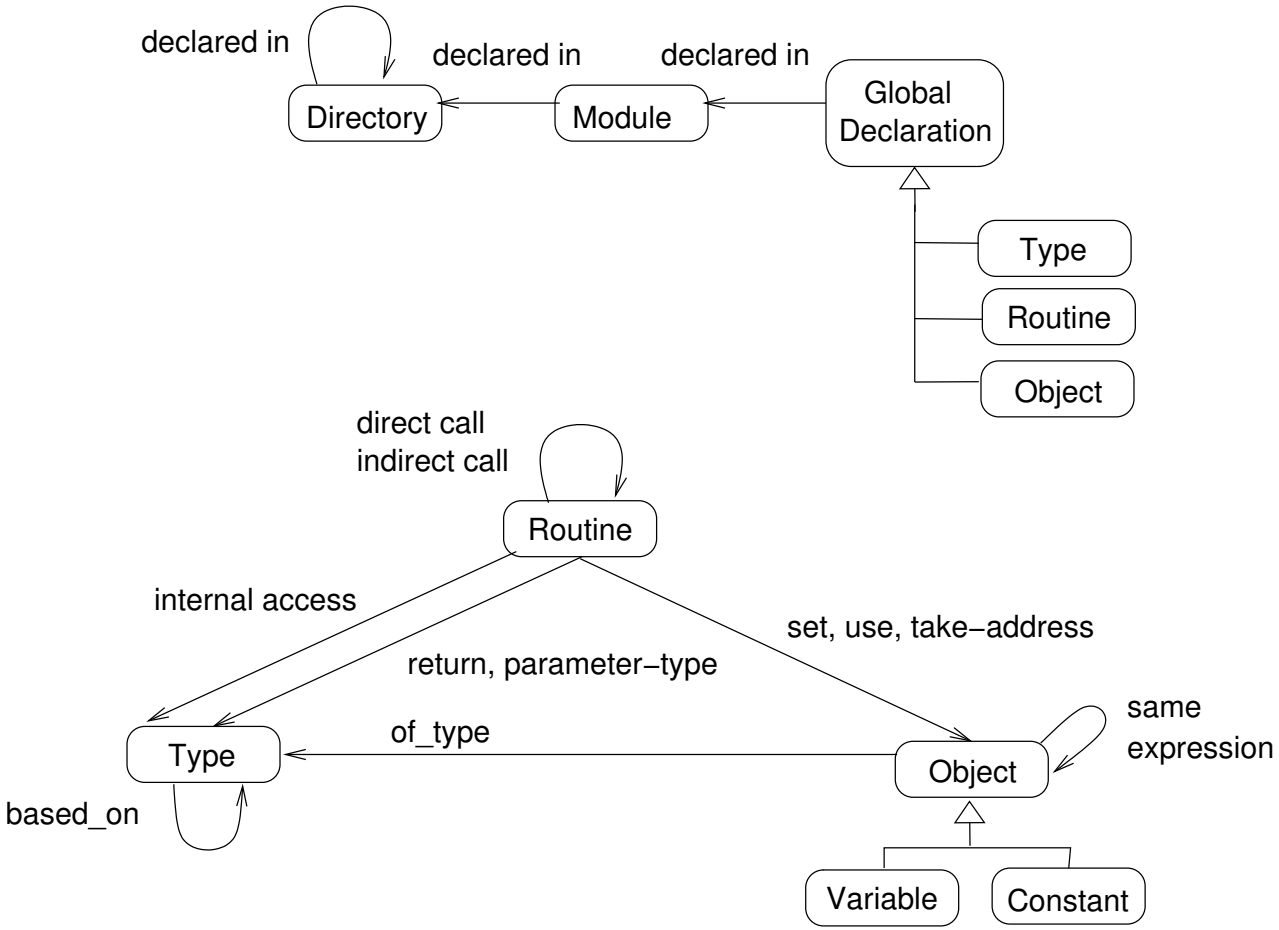


$$divergence(A, B) \Leftrightarrow \neg\exists(A', B') : (partof^*(A, A') \wedge partof^*(B, B') \wedge ref(A', B')) \wedge propagated\text{-}ref(A, B)$$

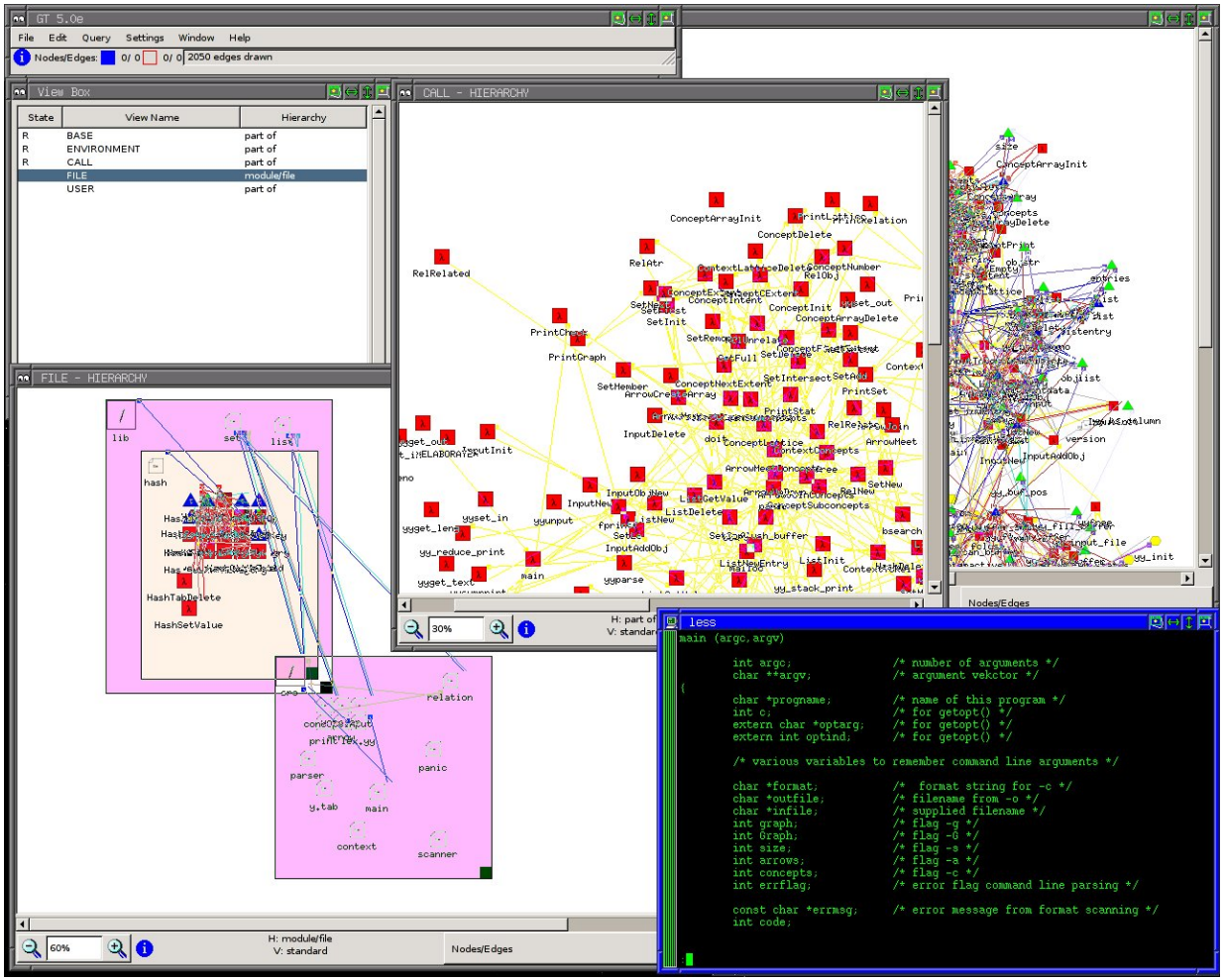
Symphony Ausführungsebene



Extraktion der Information mit Bauhaus



Knowledge Inference & Information Interpretation: Bauhaus GUI



Untersuchte Compiler

- sdcc — Small Device C Compiler
  - ANSI-C
  - Intel 8051, Zilog Z80, Dallas 80C390...
- cc1 — Teil der GNU Compiler Collection
  - (ANSI), GNU C
  - viele Plattformen

System	KLOC	C Module	Aufwand
sdcc[1]	100	49	6 h
cc1[2]	500	156	8+ h

[1] http://sourceforge.net/projects/sdcc/
[2] http://gcc.gnu.org/

Resultate für sdcc

- Abbildung von Dateien auf Module relativ einfach

Resultate für sdcc

- Abbildung von Dateien auf Module relativ einfach
- erste Iteration: viele Divergenzen

Resultate für sdcc

- Abbildung von Dateien auf Module relativ einfach
- erste Iteration: viele Divergenzen
- Verfeinerungen (5 weitere Iterationen):
  - 45 globale Deklarationen nicht in passender Übersetzungseinheit
  - die meisten in die Komponente *Global Declarations*
  - übersehene Abhängigkeiten in hypothetischen Modulsicht

Resultate für sdcc

- Abbildung von Dateien auf Module relativ einfach
- erste Iteration: viele Divergenzen
- Verfeinerungen (5 weitere Iterationen):
  - 45 globale Deklarationen nicht in passender Übersetzungseinheit
  - die meisten in die Komponente *Global Declarations*
  - übersehene Abhängigkeiten in hypothetischen Modulsicht
- Architekturverletzungen:
  - Symboltabelle referenziert den Parser
    - *declaration block number* und *line number*
  - Backend referenziert den Parser
    - globale Variable für Stackgröße von Aktivierungsblöcken

Resultate für sdcc

- Abbildung von Dateien auf Module relativ einfach
- erste Iteration: viele Divergenzen
- Verfeinerungen (5 weitere Iterationen):
  - 45 globale Deklarationen nicht in passender Übersetzungseinheit
  - die meisten in die Komponente *Global Declarations*
  - übersehene Abhängigkeiten in hypothetischen Modulsicht
- Architekturverletzungen:
  - Symboltabelle referenziert den Parser
    - *declaration block number* und *line number*
  - Backend referenziert den Parser
    - globale Variable für Stackgröße von Aktivierungsblöcken
- Architekturmuster: Optimierer referenziert Backend
  - Plattform spezifische Parameter
  - über Funktionszeiger
  - “anonyme” Abhängigkeit

Resultate für cc1

- nur Übersetzungseinheiten abgebildet (keine Verfeinerungen für globale Deklarationen)

Resultate für cc1

- nur Übersetzungseinheiten abgebildet (keine Verfeinerungen für globale Deklarationen)
- Frontend ist gut strukturiert und lose mit Middle- und Backend verbunden

Resultate für cc1

- nur Übersetzungseinheiten abgebildet (keine Verfeinerungen für globale Deklarationen)
- Frontend ist gut strukturiert und lose mit Middle- und Backend verbunden
- Backend einfach zu finden

Resultate für cc1

- nur Übersetzungseinheiten abgebildet (keine Verfeinerungen für globale Deklarationen)
- Frontend ist gut strukturiert und lose mit Middle- und Backend verbunden
- Backend einfach zu finden
- Architekturverletzungen:
  - Middleend referenziert den Präprozessor
    - verwendet Hashtabelle aus Präprozessor
    - (wir haben drei Hashtabellen im *cc1* gefunden)
  - viele Divergenzen...

Resultate für cc1

- nur Übersetzungseinheiten abgebildet (keine Verfeinerungen für globale Deklarationen)
- Frontend ist gut strukturiert und lose mit Middle- und Backend verbunden
- Backend einfach zu finden
- Architekturverletzungen:
  - Middleend referenziert den Präprozessor
    - verwendet Hashtabelle aus Präprozessor
    - (wir haben drei Hashtabellen im *cc1* gefunden)
  - viele Divergenzen...
- Middleend ist ein großer Klumpen

Erfahrungen

- Reflexion Modell muss hierarchische Architekturen unterstützen
- Anwendungswissen bei der Erstellung der hypothetischen Architektur wesentlich
- manuelle Schritte aufwändig, Iterationen üblich
- Toolunterstützung zur Erstellung der Abbildung
- dann aber automatisierte Anwendung für Folgeiterationen
- Abgleich von Soll- gegen Ist-Architektur