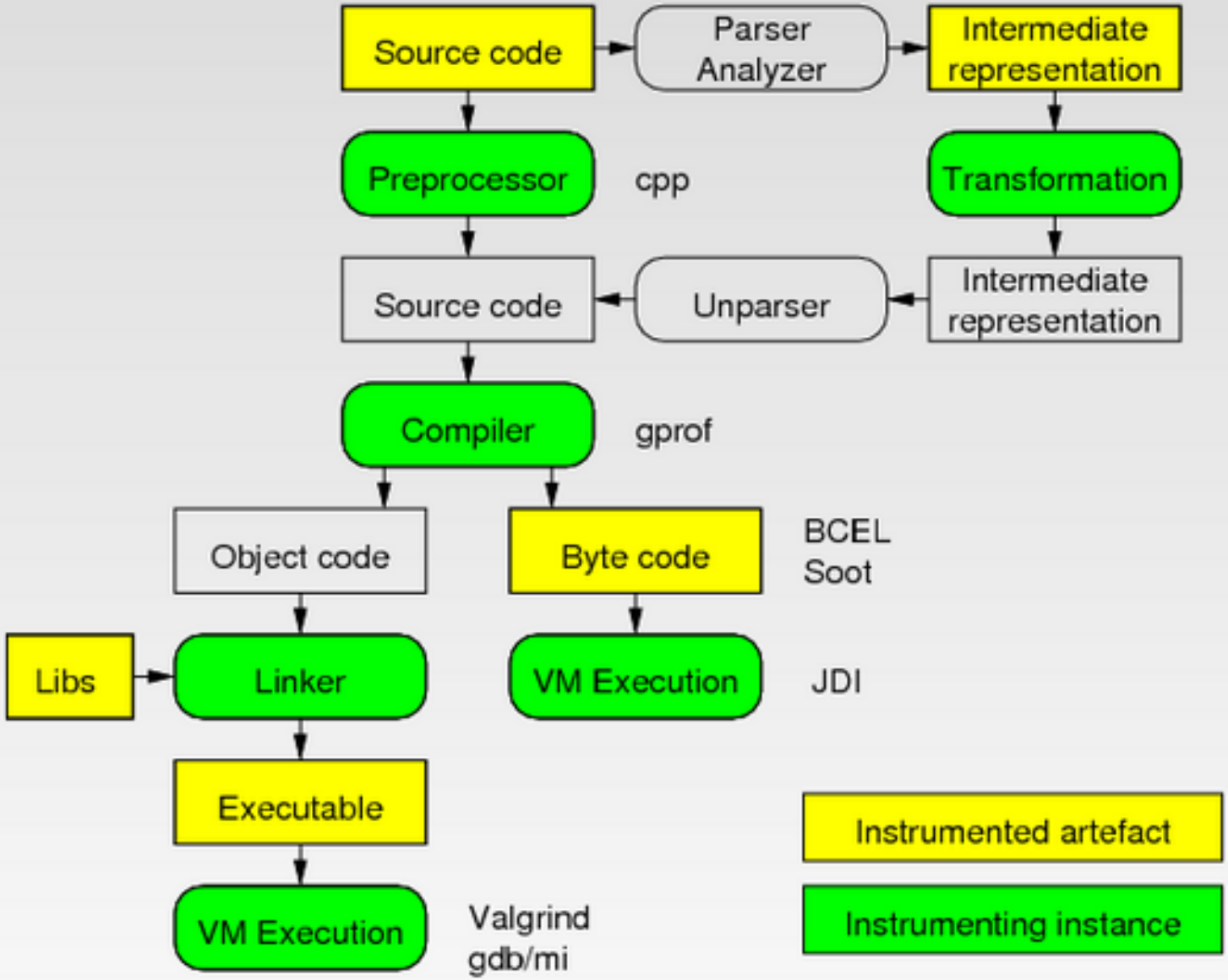
Instrumentierung



Instrumentierung

- Performanz?
  - Debug-Schnittstelle i.d.R. langsam
  - Profiler schnell
- Aufwand?
  - Profile-, Coverage-Information umsonst
  - automatisierbar?
- Bezug zum Quellcode?
  - schwierig bei Maschinencode
  - unabhängig von Compiler/Prozessor?
- Unabhängig von Programmiersprache?

Datenmenge

Problem häufig: zu viele Daten!

Lösungen:

- selektive Instrumentierung
- sofortige Verarbeitung (online)
- Umkodierung
  - verlustfreie Kompression
  - Filterung
  - Vorverarbeitung (Trace Summarization)

Auswertung

- Aggregation
  - Statistiken
- Inferenz: Ableitung neuen Wissens
  - maschinelles Lernen
- Visualisierung

je online/offline möglich

Beispiel: Profiling

Ziel: Performance-Engpässe finden

Techniken:

- Messen der Zeiten
  - Zeit vor/nach Ausführung
  - Mitzählen, Summieren
- PC Sampling

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
33.34	0.02	0.02	7208	0.00	0.00	open
16.67	0.03	0.01	244	0.04	0.12	offtime
16.67	0.04	0.01	8	1.25	1.25	memccpy
16.67	0.05	0.01	7	1.43	1.43	write
...						

Beispiel: Dynamisches Slicing (Agrawal und Horgan 1990)

Statisches Slicing: Erreichbarkeit im System-Dependency-Graph

Dynamisches Slicing: Verfolgen von Werten zur Laufzeit

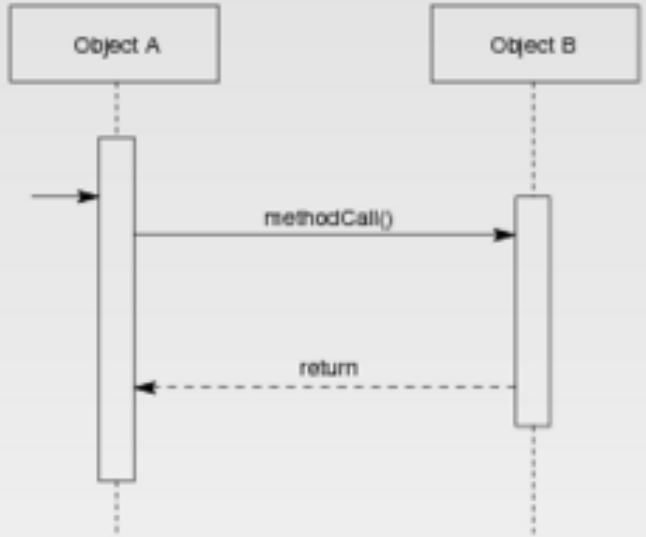
Jeder Wert wird annotiert:

- wo wurde er berechnet?
- aus welchen Werten?

Dann: Rückwärtsverfolgung der Verweise

Beispiel: Collaborations (Richner und Ducasse 2002)

- Aufzeichnen von Methodenaufrufen
  - Aufrufer, Aufgerufener (Klasse, Instanz), Methode
- Pattern-Matching
  - Suchen ähnlicher Muster
  - Klassen, Instanzen, Methoden, Struktur
- Auswahl über Klassen/Methoden
  - Collaboration Browser
- Visualisierung als Sequenzdiagramm



Beispiel: Invariantenerkennung (Ernst 2000)

```
public class StackAr {
  Object[] theArray;
  int topOfStack;
  ...
}
```

Invarianten:

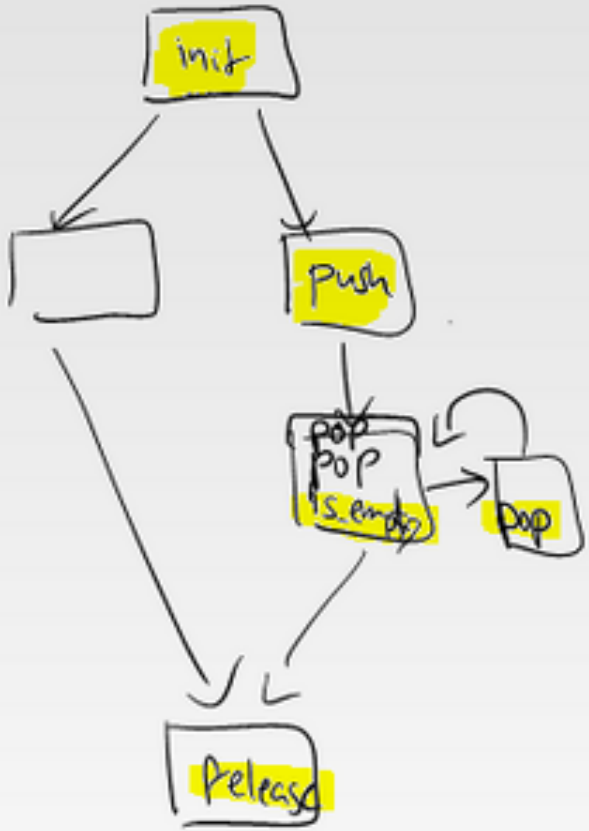
- `theArray != null`
- `theArray.getClass() == java.lang.Object[].class`
- `topOfStack >= -1`
- `topOfStack <= theArray.length - 1`
- `theArray[0..topOfStack] elements != null`
- `theArray[topOfStack+1..] elements == null`

Beispiel: Invariantenerkennung (Ernst 2000)

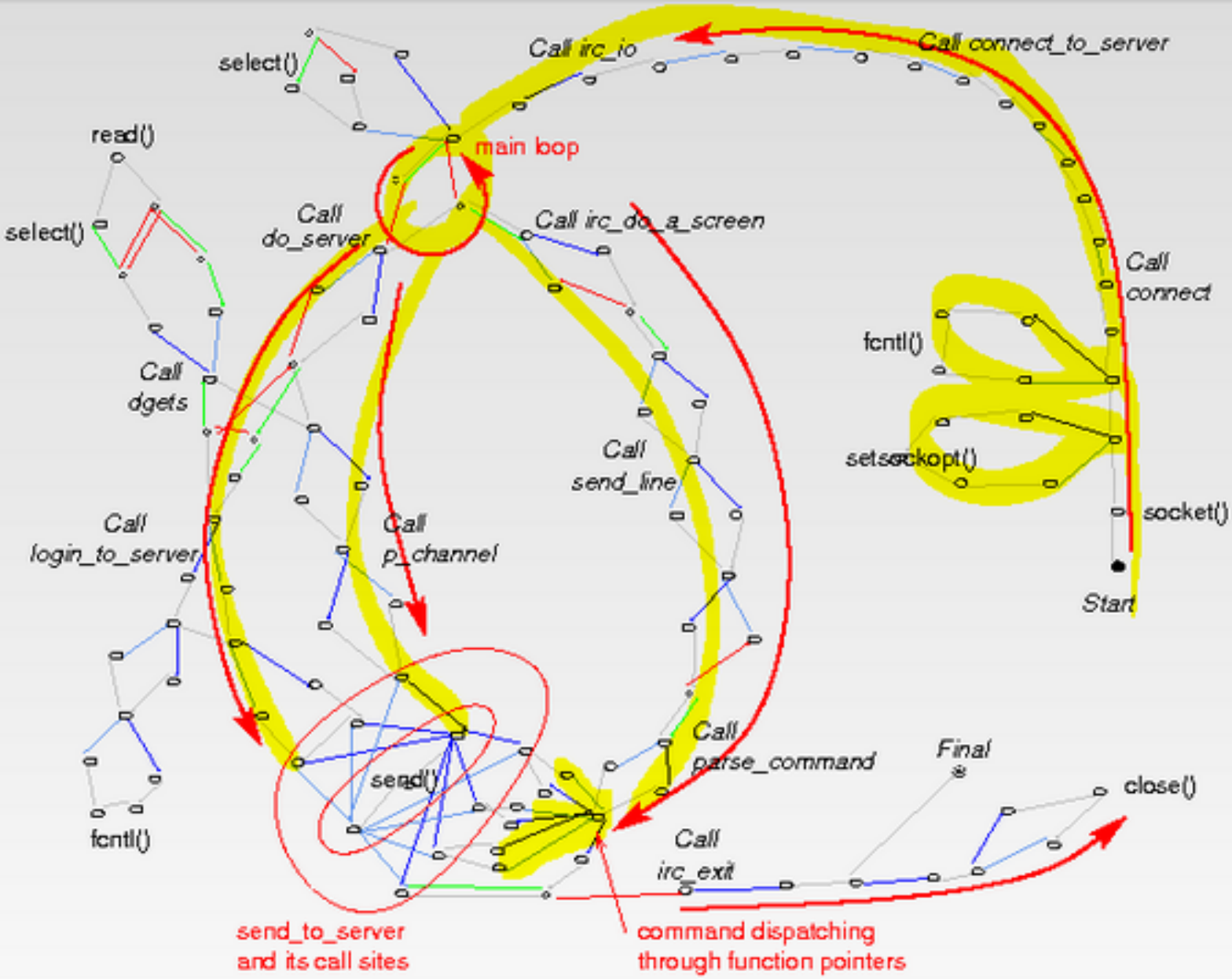
- Betrachtung aller Variablen
- auch abgeleitete Variablen
  - sum, min, max
  - first, second, last
  - s[i], s[i-1], Teilsequenzen s[0..i], s[0..i-1]
  - Anzahl Aufrufe einer Funktion
- Instanziierung und Testen aller potentiellen Invarianten
- Filterung der übrigen Invarianten
  - hinreichend signifikant?

Beispiel: Dynamic Object Process Graph (Quante und Koschke 2006)

- Rekonstruktion des Kontrollflussgraphen
- Projektion auf ein Objekt
- Instrumentierung von
  - Verzweigungen
  - Routinenaufrufen
  - Operationen auf Objekt
- Verwandt zu Slicing



Beispiel: Dynamic Object Process Graph

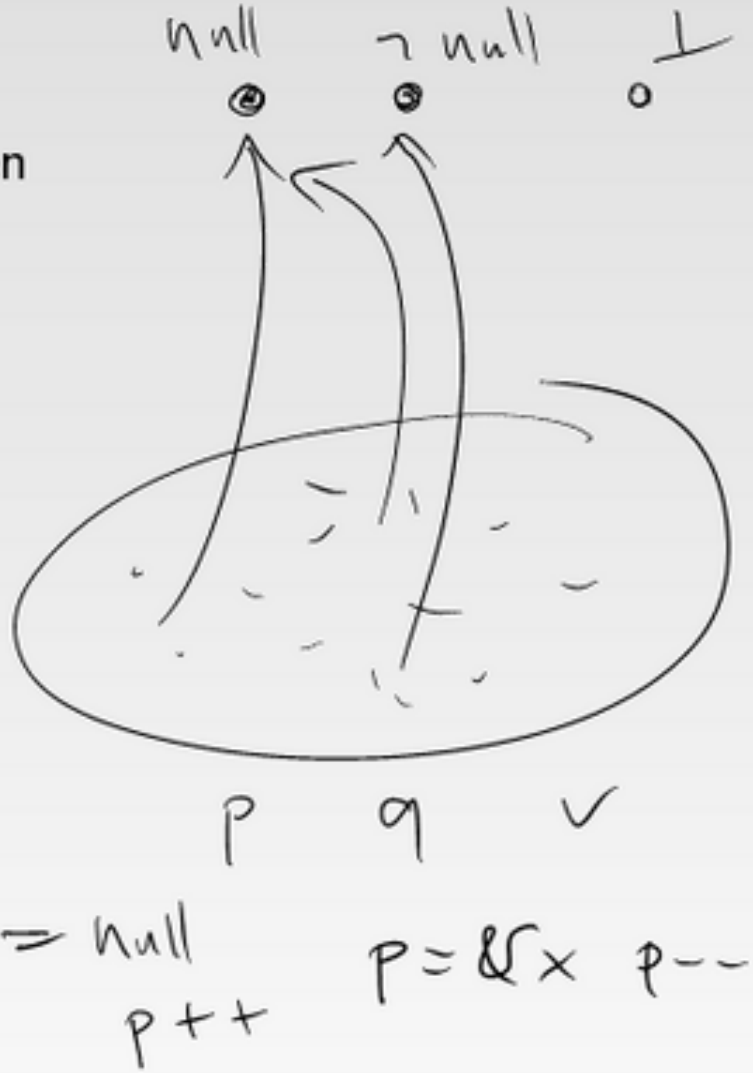


Statische Analyse von Programmverhalten

- Analyse des Codes
- Modell des Programmzustands erstellen
- Mögliches Verhalten folgern
  - abstrakte Ausführung

Abstrakte Ausführung:

- meist Datenflussanalyse
- Transferfunktion für jede Anweisung
  - Änderung des abstrakten Zustands?
- Beispiel: `y = x++;`



Wahl der abstrakten Domäne

- Domäne: {even, odd, either}
  <x odd, y either> `y = x++;` <x even, y odd>
  <x even, y either> `y = x++;` <x odd, y even>
- Domäne: {prime, nonprime, anything}
  <x prime, y anything> `y = x++;` <x anything, y prime>
- Domäne: Menge von möglichen nat. Zahlen
  <x $\in \{3,5,7\}$, y $\in \mathbb{N}$> `y = x++;` <x $\in \{4,6,8\}$, y $\in \{3,5,7\}$>
- Domäne: symbolische Auswertung
  <x_n, y_n> `y = x++;` <$x_{n+1} = x_n + 1$, $y_{n+1} = x_n$>

Wahl der abstrakten Domäne

Abstraktion bestimmt

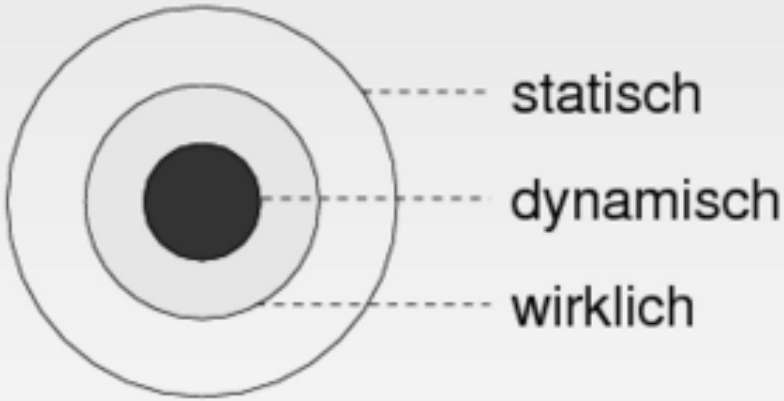
- Kosten (Zeit und Raum)
- Genauigkeit (Informationsverlust)

Statisch vs. Dynamisch

Statisch	Dynamisch
abstrakte Domäne aufwändig wenn genau	konkrete Ausführung kann aufwändig sein
konservativ wegen Abstraktion	präzise keine Abstraktion
vollständig (plus mehr) da konservativ	unvollständig nicht verallgemeinerbar
begrenzte Sichtweite eher lokal	kann entfernte Beziehungen aufdecken
Abstraktion bestimmt Kosten, Genauigkeit	Testsuite bestimmt Kosten, Genauigkeit

Statisch vs. Dynamisch

- dynamisch: zu konkret, nicht verallgemeinerbar
  $\longrightarrow$ Teilmenge
- statisch: zu abstrakt, daher ungenau
  $\longrightarrow$ Obermenge
- Wahrheit liegt (meist) dazwischen



Statisch vs. Dynamisch

Beispiel:

```
y = f(x);
if (y < 0)
    y = -y;
```

Dynamisch:	$y \in \{0, 2, 7, 43, 79\}$
Statisch:	$0 \leq y \leq$ `MAX_INT`
Wirklich:	$0 \leq y \leq 100$

Kombination

- Aggregation
  - dynamisch $\rightarrow$ statisch: Übergeneralisierung erkennen
  - statisch $\rightarrow$ dynamisch: gezielte Instrumentierung
- Hybride Analysen

Beispiele – Übersicht

Anwendung	dynamisch	statisch
Typprüfung	×	×
Speicherzugriffsprüfung	×	×
Slicing	×	×
Merkmalsuche	×	×
Protokollerkennung	×	×
Protokollverifikation	×	×
Metriken	Verhalten	Struktur
- Profiling	×	—
- Testabdeckung	×	—
Finden toten Codes	—	×
Klonerkennung	—	×
Zeigeranalyse	leicht	schwierig

Formale Begriffsanalyse

2. Begriffsanalyse
   - Formaler Kontext
   - Ordnung
   - Begriffsverband
   - Supremum und Infimum
   - Verkürzter Verband