Polarisierte Untertypisierung höherer Ordnung

Martin Steffen

12. November 1998

Inhalt

- Motivation
- Typen für Objektorientierung
- “F-omega-sub”
- Entscheidbarkeit
- Ausblick

typisierte Programmiersprachen

- „semantische“ Phase des Compilers:

  Lexer ⇝ Parser ⇝ Typsystem ⇝ „oper. Semantik“

  regulär kontextfrei ? unentscheidbar

- starke Typsicherheit:

  wohltypisierte Programme sind frei von Laufzeitfehlern

- am besten: statisch (⇒ Effizienz)

parametrische Polymorphie

- polymorph: ein Programm kann mehrere Typen haben
- Beispiel: Vertauschen

$$swap(x{:}\mathbb{N}, y{:}\mathbb{B}) \quad = \quad (y, x) \quad : \quad \mathbb{N} \times \mathbb{B} \to \mathbb{B} \times \mathbb{N}$$

- besser: eine generische $swap$-Funktion für alle Typen

$$swap : \forall X, Y.X \times Y \to Y \times X$$
$$swap\ X\ Y\ (x{:}X, y{:}Y) = (y, x)$$

$\Rightarrow$ **parametrische/universelle** Polymorphie

Untertypen

Ist S ein **Untertyp** von T, dann kann ein Programm des Typs S anstelle von Programmen des Typs T verwendet werden

$\Rightarrow$ Ordnung auf den Typen ($\leq$)

- intuitiv: Teilmengen $S \subseteq T$, z.B. $Int \leq Real$.
- Kombination mit universeller Polymorphie:

$$list\_max : \forall X \leq Ord.(List\ of\ X) \to X$$

- beschränkt-universelle Quantifikation

Typoperatoren

- Beispiel 1: „$\boldsymbol{List\ of\ \_}$“ ist kein Typ, “$\boldsymbol{List\ of}\ \mathbb{N}$“ schon, z.B.

$$[4, 5, 0] : \boldsymbol{List\ of}\ \mathbb{N}$$

$\Rightarrow$ Typ**operator** = Funktion von Typen nach Typen

- Beispiel 2: Signatur/Methodenschnittstelle eines Objektes

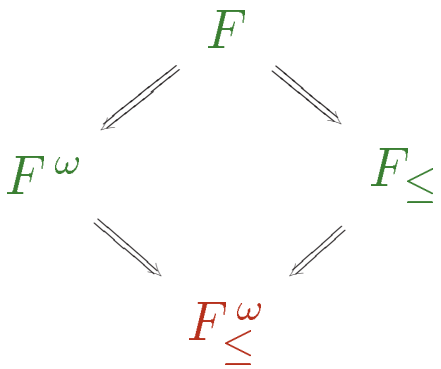
$$\begin{aligned} PointSig\ \ of\ X = \{\ & getx : X \to \mathbb{N}, \\ & setx : X \to \mathbb{N} \to X\ \} \end{aligned}$$

Wunschliste (Forts.)

- Polymorphie
  - universelle Polymorphie
  - Untertypen
- Funktionen höherer Ordnung
- Kapselung von Objekten
- Vererbung
- späte/dynamische Methodenbindung
- ...

formale Modelle: typisierte λ-Kalküle

[Landin, 1966] „The next 700 programming languages“



- F: der polymorphe λ-Kalkül [Girard, 1971] [Reynolds, 1974]
- $F_\leq$: [Cardelli and Wegner, 1985] . . .
- F^ω [Girard, 1971]
- $F_\leq^\omega$ [Cardelli, 1990] [Mitchell, 1990] . . .

Weiteres Vorgehen

- Festlegung der Syntax
- Axiomatisierung statischer Eigenschaften
  - Wann besitzt ein Programm t den Typ T?
  - Wann ist ein Typ S ein Untertyp von T?

$\Rightarrow$ formales Deduktionssytem

Syntax für $F^{\omega}_{\leq}$

- drei Ebenen: Programme, Typen und Arten

t	$::=$	$x \mid fun(x{:}T)t \mid t\ t$	Funktionen
	$\mid$	$fun(X \leq T)t \mid t\ T$	universelle Polymorphie, $\leq$
T	$::=$	$T \to T$	Funktionen
	$\mid$	$All(X \leq T)T \mid X$	universelle Polymorphie, $\leq$
	$\mid$	$Top(K)$	
	$\mid$	$Fun(X{:}K)T \mid T\ T$	Typoperatoren
K	$::=$	$\star \mid K \to K$	Arten (= Typen der Typen)

Urteile

- Bisher: nur Syntax, es fehlen die Zusammenhänge, insbesondere

$\Rightarrow$ Urteile z.B.:

$\Gamma \vdash t : T$	Programm t ist vom Typ T
$\Gamma \vdash S \leq T$	S ist Untertyp von T
$\Gamma \vdash T : K$	Typ T besitzt Art K

$F^{\omega}_{\leq}$: Untertypsystem

- Axiomatisierung von $\leq$ mittels Deduktionsregeln
- zwei Arten von Regeln
  1. sprachunabhängige Eigenschaften von $\leq$, z.B. Transitivität.

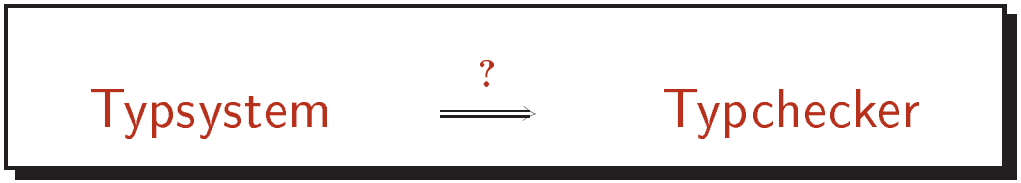
$$\frac{\Gamma \vdash \boldsymbol{S} \leq \boldsymbol{U} \quad \Gamma \vdash \boldsymbol{U} \leq \boldsymbol{T}}{\Gamma \vdash \boldsymbol{S} \leq \boldsymbol{T}} \qquad \text{(S-Trans)}$$

  2. strukturell, z.B. für $\rightarrow$–Typen

$$\frac{\Gamma \vdash \boldsymbol{T_1} \leq \boldsymbol{S_1} \qquad \Gamma \vdash \boldsymbol{S_2} \leq \boldsymbol{T_2}}{\Gamma \vdash \boldsymbol{S_1} \rightarrow \boldsymbol{S_2} \leq \boldsymbol{T_1} \rightarrow \boldsymbol{T_2}} \qquad \text{(S-Arrow)}$$

Ziel

- Gegeben: Spezifikation der (Unter–)Typsysteme
- Gesucht: Algorithmus zur Überprüfung der Urteile.



Hauptprobleme

strukturelle Regeln $\to$, $\forall$ ... (fast) problemlos

$$\frac{\Gamma \vdash \boldsymbol{T_1} \leq \boldsymbol{S_1} \qquad \Gamma \vdash \boldsymbol{S_2} \leq \boldsymbol{T_2}}{\Gamma \vdash \boldsymbol{S_1} \to \boldsymbol{S_2} \leq \boldsymbol{T_1} \to \boldsymbol{T_2}}$$

nicht jedoch

1. Transitivität:

$$\frac{\Gamma \vdash \boldsymbol{S} \leq \boldsymbol{U} \quad \Gamma \vdash \boldsymbol{U} \leq \boldsymbol{T}}{\Gamma \vdash \boldsymbol{S} \leq \boldsymbol{T}}$$

2. Konversion

$$\frac{S =_\beta S' \qquad \Gamma \vdash \boldsymbol{S'} \leq \boldsymbol{T'} \qquad T' =_\beta T}{\Gamma \vdash \boldsymbol{S} \leq \boldsymbol{T}}$$

Transitivität

- Ziel: Transitivitätsregel ist überflüssig = „Schnittelimination“

$$\frac{U_1 \leq S_1 \quad S_2 \leq U_2}{S_1 \to S_2 \leq U_1 \to U_2} + \frac{T_1 \leq U_1 \quad U_2 \leq S_2}{U_1 \to U_2 \leq T_1 \to T_2}$$

$$\frac{\dfrac{T_1 \leq U_1 \quad U_1 \leq S_1}{T_1 \leq S_1} \qquad \dfrac{S_2 \leq U_2 \quad U_2 \leq T_2}{S_2 \leq T_2}}{S_1 \to S_2 \leq T_1 \to T_2}$$

- Probleme:
  - nur beweisbar auf Typen in Normalform
  - kann die Normalform essentiell zerstören

Ergebnisse (I)

Theorem. *Das Untertypproblem* $\Gamma \vdash S \leq T$ *für* $F^{\omega}_{\leq}$ *ist entscheidbar.*

Satz. *Jedes wohltypisierte Programm hat einen minimalen Typ.*

Korrollar. *Das Typisierungsproblem* $\Gamma \vdash t : T$ *für* $F^{\omega}_{\leq}$ *ist entscheidbar.*

[Steffen and Pierce, 1994]

$F^{\omega}_{\leq}$ als OO-Kalkül

- [Hofmann and Pierce, 1995]: $F^{\omega}_{\leq}$ als Basis für OO-Sprachen
  - klassenbasiert
  - Einfachvererbung
  - Kapselung

  vorausgesetzt:

  Signatur/Methodenschnittstelle = monotoner Typoperator

$$PointSig = Fun(X).\{\ getx : X \to \mathbb{N},\ setx : X \to \mathbb{N} \to \boldsymbol{X}\ \}$$

- Klassenvererbung $\Rightarrow$ Untertypbeziehung zwischen Instanzen
- Abwesenheit von binären Methoden

$F^{\omega}_{\leq}$ + Monotonie

– Erweiterung von $F^{\omega}_{\leq}$ um Monotonieinformation ⇒

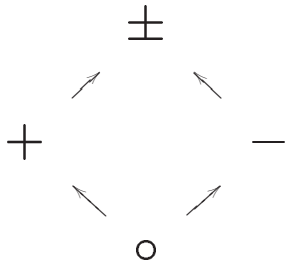
„polarisiertes $F^{\omega}_{\leq}$“

Wenn $Int \leq Real$,

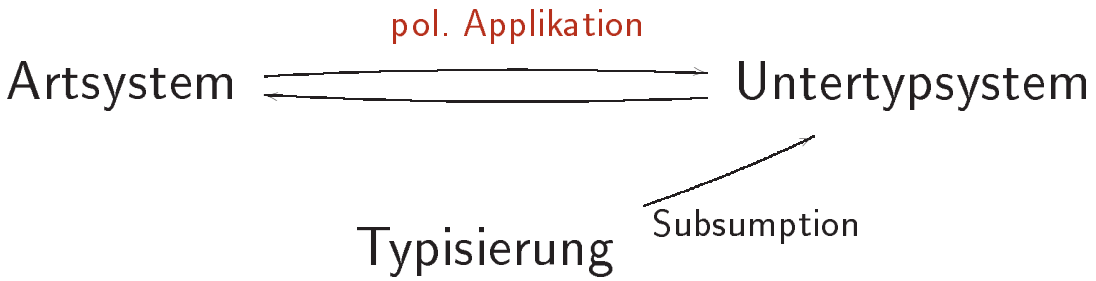
dann $\boldsymbol{List\ of}\ Int \leq \boldsymbol{List\ of}\ Real$?

$$\frac{\ldots \boldsymbol{X_1} \leq \boldsymbol{X_2} \ldots \vdash T\ X_1 \leq T\ X_2}{\Gamma \vdash T\ \in\ K_1 \to^{+} K_2}$$

- Insgesamt vier Polaritäten ⇒ Subkinding



Analyse und Lösungsansatz



- Zerbrechen der direkten Abhängigkeit von Untertypisierung und Kinding $\Rightarrow$ „Stratifizierung“
- Terminierung
- Verallgemeinerung der $\forall$-Untertypregel

Ergebnisse (II)

Theorem. *Das Untertypproblem $\Gamma \vdash S \leq T : K$ und das Kindingproblem $\Gamma \vdash T : K$ für polarisiertes $F^{\omega}_{\leq}$ sind entscheidbar.*

Satz. *Jedes wohltypisierte Programm hat einen minimalen Typ.*

Korrollar. *Das Typisierungsproblem $\Gamma \vdash t : T$ für polarisiertes $F^{\omega}_{\leq}$ ist entscheidbar.*

Ausblick & weitere Arbeiten

- Modell, am einfachsten PER-Modell
- Operatoren mit beschränkten Argumenten ([Compagnoni and Goguen, 1997]):

$$Fun(X \leq S)T \qquad \text{statt} \qquad Fun(X\colon K)T$$

- „lokale“ Typinferenz (z.B. [Pierce and Turner, 1998] für $F_{\leq}$, in Pict)

Literatur

[Cardelli, 1990] Cardelli, L. (1990). Notes about $F^{\omega}_{<:}$. Unpublished manuscript.

[Cardelli and Wegner, 1985] Cardelli, L. and Wegner, P. (1985). On understanding types, data abstraction and polymorphism. *Computing Surveys*, 17(4):471–522.

[Compagnoni and Goguen, 1997] Compagnoni, A. and Goguen, H. (1997). Typed operational semantics for higher order subtyping. Technical Report ECS-LFCS-97-361, Department of Computer Science, University of Edinburgh. Submitted for publication in *Information and Computation*.

[Girard, 1971] Girard, J.-Y. (1971). Une extension de l'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In Fenstad, J. E., editor, *Second Scandinavian Logic Symposium '71 (Oslo, Norway)*, number 63 in Studies in Logic and the Foundations of Mathematics, pages 63–92. North-Holland.

[Gunter and Mitchell, 1994] Gunter, C. A. and Mitchell, J. C. (1994). *Theoretical Aspects of Object-Oriented Programming, Types, Semantics, and Language Design*. Foundations of Computing Series. MIT Press.

[Hofmann and Pierce, 1995] Hofmann, M. and Pierce, B. (1995). A unifying type-theoretic framework for objects. *Journal of Functional Programming*, 5(4):593–635. Previous versions appeared in the Symposium on Theoretical Aspects of Computer Science, 1994, (pages 251–262) and, under the title "An Abstract View of Objects and Subtyping (Preliminary Report)," as University of Edinburgh, LFCS technical report ECS-LFCS-92-226, 1992.

[Landin, 1966] Landin, P. J. (1966). The next 700 programming languages. *Communications of the ACM*, 9(3):157–166.

[Mitchell, 1990] Mitchell, J. C. (1990). Toward a typed foundation for method specialization and inheritance. In *Seventeenth Annual Symposium on Principles of Programming Languages (POPL) (San Fancisco, CA)*, pages 109–124. ACM. Also in the collection [Gunter and Mitchell, 1994].

[Pierce and Turner, 1998] Pierce, B. C. and Turner, D. N. (1998). Local type inference. In *Proceedings of POPL '98*. ACM. Also as Indiana University Technical Report CSCI TR #493.

[Reynolds, 1974] Reynolds, J. (1974). Towards a theory of type structure. In Robinet, B., editor, *Colloque sur la programmation (Paris, France)*, volume 19 of

Lecture Notes in Computer Science, pages 408–425. Springer-Verlag.

[Steffen and Pierce, 1994] Steffen, M. and Pierce, B. (1994). Higher-order subtyping. In Olderog, E.-R., editor, *Proceedings of PROCOMET '94*, pages 511–530. IFIP, North-Holland. Full version in *Theoretical Computer Science*, vol. 176, no. 1–2, pp. 235–282, 1997.