Software Engineering: WG Programming Languages: Cass's projects

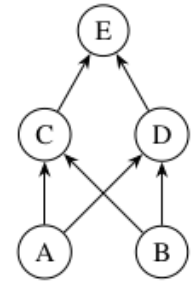
Cass Alexandru

2026-04-29

The Problem

- Lecture "Concepts of Programming"
- Typical Exam Exercise:
  - Give a graph showing subtyping relationships between type atoms A,B,C,...
    1. Given a type, give a type that is related by one of $<, ||, >$ to it
    2. Given two types, state which of $<, ||, >$ their relation is
  - ... for type atoms or *compound types* built from atoms with the product $*$ or arrow (function) -> type constructors.

Es gelten die Untertypbeziehungen A $\preccurlyeq$ C, A $\preccurlyeq$ D, B $\preccurlyeq$ C, B $\preccurlyeq$ D, C $\preccurlyeq$ E und D $\preccurlyeq$ E, die in der nebenstehenden Abbildung visualisiert sind.



In der folgenden Tabelle werden je zwei Typen in Relation gesetzt:

- $t_1 \prec t_2$ bedeutet, dass t_1 ein Untertyp von t_2 ist ($t_1 \preccurlyeq t_2$), nicht jedoch t_2 ein Untertyp von t_1.
- $t_1 \succ t_2$ bedeutet, dass t_2 ein Untertyp von t_1 ist ($t_2 \preccurlyeq t_1$), nicht jedoch t_1 ein Untertyp von t_2.
- $t_1 \parallel t_2$ bedeutet, dass t_1 und t_2 unvergleichbar sind, das heißt, es gilt weder $t_1 \preccurlyeq t_2$ noch $t_2 \preccurlyeq t_1$.

Füllen Sie die Lücken in der Tabelle aus. In der ersten und dritten Spalte müssen Sie **einen** Typ eintragen, in der zweiten Spalte eine Relation ($\prec$ oder $\succ$ oder $\parallel$).

Für richtige Antworten (ganze Lücke) erhalten Sie zwei Punkte, für falsche Antworten werden zwei Punkte abgezogen. Nicht ausgefüllte Zeilen wirken sich nicht auf die Punktzahl aus. Diese Aufgabe wird mit mindestens 0 Punkten bewertet.

Zur Erinnerung: Die folgenden Deduktionsregeln gelten für die Relation »$\preccurlyeq$«:

$$\frac{t_1 \preccurlyeq t_2 \qquad t_2 \preccurlyeq t_3}{t_1 \preccurlyeq t_3} \qquad \frac{t_1 \preccurlyeq t_1' \qquad t_2 \preccurlyeq t_2'}{t_1 * t_2 \preccurlyeq t_1' * t_2'} \qquad \frac{t_1' \preccurlyeq t_1 \qquad t_2 \preccurlyeq t_2'}{t_1 \to t_2 \preccurlyeq t_1' \to t_2'}$$

t_1	Relation	t_2
C	$\succ$	A
C $*$ A	$\prec$	C $*$ C, C $*$ D, C $*$ E, E $*$ A, E $*$ C, E $*$ D, E $*$ E
A $*$ E	$\parallel$	B $*$ D
C $\to$ C	$\succ$	E $\to$ C, C $\to$ A, C $\to$ B, E $\to$ B, E $\to$ A
(D $*$ B) $\to$ E	$\prec$	(B $*$ B) $\to$ E, (A $*$ B) $\to$ E
A $\to$ C	$\parallel$	D $\to$ E
C $\to$ D	$\prec$	A $\to$ E
E $\to$ A	$\parallel$	Es gibt unendlich viele Lösungen, z.B. A. Falsch sind alle Typen der Form $\square \to \square$
D $\to$ B $\to$ E	$\prec$	B $\to$ B $\to$ E, A $\to$ B $\to$ E
(A $\to$ D) $\to$ B	$\succ$	(A $\to$ E) $\to$ B

Your Task

- Generate random graphs for subtyping relationships between type atoms
- Given such a graph, generate random (compound) types for exercises of form 1 and 2.
- Possible extension (dep. on team size): write a browser game version of this which students can practice with

The Problem

Syntax-Highlighted Haskell code:

```haskell
insert :: L OList -> OList
insert FNil = Nil
insert (ConsF a Nil) = Cons a Nil
insert (ConsF a (Cons b r'))
   |  a <= b     = Cons a (Cons b r')
   |  otherwise  = Cons b (insert (ConsF a r'))
```

Syntax-Highlighted Agda version of equivalent code:

```agda
insert : L OList -> OList
insert FNil = Nil
insert (ConsF a Nil) = Cons a Nil
insert (ConsF a (Cons b r')) with a ≤?≥ b
...| inl a≤b  = Cons a (Cons b r')
...| inr b≥a      = Cons b (insert (ConsF a r'))
```

- Haskell syntax highlighting uses only lexing information
- This means no distinction between types, type constructors & constraints, identifier & variable names

Your task

- Develop a tool, as a plugin for GHC or reading its interface files, to output information relevant for syntax highlighting.
- Ideally several backends
  - MVP (minimum viable product): LaTeX
  - language server protocol (for integration in syntax-highlighting within IDEs)
  - hyperlinked & highlighted html
  - Typst

The Problem

- `.spd` format: Samsung S-Note format
- zip-archive containing binary `.page` files
- handwriting internally stored as vector graphics
- However: Export only possible to rasterized formats, losing information and blowing up file size

Your Task

- Reverse-engineer the binary `.page` file format to extract the vector graphic information
- This may involve:
  - decompiling the S-Note apk
  - analyzing its library dependencies
  - running it in a VM with file-watches, binary diffs etc
- Success is not guaranteed, but you can document your findings & process