Formal typesytem description

Ferdinand van Walree(3874389) and Matthew Swart(5597250)

November 29, 2015

0.1 Typesystem

In this document we define a formal specification of the types and typerules of our typesystem. The typerules describe what combinations of types are ill-typed. That is, if we were to encounter those types in our program, then our program would produce an error. We have chosen to leave out all well-typed rules, because we felt these weren't necessary, afterall we do not ever check whether a specific combination of types is allowed. We only check whether such combinations are not allowed.

We aren't too familiar with writing proofs such as you will find in this document, therefore we have also taken some liberty in writing the proofs. In otherwords, the proofs might not be well written, but they should be understandable.

0.2 TyCons

TyCons represents the type that a diag can be. First we'll define the different TyCons, then we'll define the TyCons for each diagram.

$$\frac{}{\Gamma \Vdash \text{Prog} : \text{TyCons}} \text{(t-ProgTyCons)}$$

$$\frac{}{\Gamma \Vdash \text{Interp} : \text{TyCons}} \text{(t-InterpTyCons)}$$

$$\frac{}{\Gamma \Vdash \text{Comp} : \text{TyCons}} \text{(t-CompTyCons)}$$

$$\frac{}{\Gamma \Vdash \text{PlatF} : \text{TyCons}} \text{(t-PlatfTyCons)}$$

$$\frac{}{\Gamma \Vdash \text{Executed} : \text{TyCons}} \text{(t-ExecutedTyCons)}$$

$$\frac{}{\Gamma \Vdash \text{Compiled} : \text{TyCons}} \text{(t-CompiledTyCons)}$$

$$\frac{}{\Gamma \Vdash \text{Framework} : \text{TyCons}} \text{(t-FrameworkTyCons)}$$

$$\frac{}{\Gamma \Vdash \text{Runnable} : \text{TyCons}} \text{(t-RunnableTyCons)}$$

Some TyCons also have a TyCons type

$$\frac{}{\Gamma \Vdash \text{Prog} : \text{Runnable}} \text{(tyCons-prog-runn)}$$

$$\frac{}{\Gamma \Vdash \text{Interp} : \text{Runnable}} \text{(tyCons-interp-runn)}$$

$$\frac{}{\Gamma \Vdash \text{Comp} : \text{Runnable}} \text{(tyCons-comp-runn)}$$

$$\frac{}{\Gamma \Vdash \text{PlatF : Framework}}\ \text{(tyCons-platf-frame)}$$

$$\frac{}{\Gamma \Vdash \text{Interp : Framework}}\ \text{(tyCons-Interp-frame)}$$

The following proofs determine the TyCons for each diagram

$$\frac{}{\Gamma \Vdash \text{Program p l : Prog}}\ \text{(tyCons-prog)}$$

$$\frac{}{\Gamma \Vdash \text{Interper i l m : Interp}}\ \text{(tyCons-interp)}$$

$$\frac{}{\Gamma \Vdash \text{Compilerc l1 l2 ml : Comp}}\ \text{(tyCons-comp)}$$

$$\frac{}{\Gamma \Vdash \text{Platform m : Platf}}\ \text{(tyCons-platf)}$$

$$\frac{}{\Gamma \Vdash \text{Execute d1 d2 : Executed}}\ \text{(tyCons-execute)}$$

$$\frac{}{\Gamma \Vdash \text{Compile d1 d2 : Compiled}}\ \text{(tyCons-compile)}$$

0.3 Recursive

The recursive datatype is used to represent whether a Compile is recursively nested within another compile. Given a compile of the following form: compile L with R end. If there was a Compile in L, then it would be left recursive. If there was a compile in R, then it would be right recursive. If there are no Compiles, then they are not recursive.

$$\frac{}{\Gamma \Vdash \text{Not\_recursive : Recursive}}\ \text{(recurse-not)}$$

$$\frac{}{\Gamma \Vdash \text{Left\_recursive : Recursive}}\ \text{(recurse-left)}$$

$$\frac{}{\Gamma \Vdash \text{Right\_recursive : Recursive}}\ \text{(recurse-right)}$$

$$\frac{}{\Gamma \Vdash \text{Program p l : Not\_recursive}}\ \text{(recurse-prog)}$$

$$\frac{}{\Gamma \Vdash \text{Interpreter i l m : Not\_recursive}}\ \text{(recurse-interp)}$$

$$\frac{}{\Gamma \Vdash \text{Compiler c l1 l2 m} : \text{Not\_recursive}}\ \text{(recurse-comp)}$$

$$\frac{}{\Gamma \Vdash \text{Platform m} : \text{Not\_recursive}}\ \text{(recurse-plat)}$$

$$\frac{}{\Gamma \Vdash \text{Execute d1 d2} : \text{Not\_recursive}}\ \text{(recurse-execute)}$$

$$\frac{\Gamma \Vdash \text{Compile d1 d2}}{\Gamma \Vdash \text{d1} : \text{Left\_recursive}}\ \text{(recurse-compile)}$$

$$\frac{\Gamma \Vdash \text{Compile d1 d2}}{\Gamma \Vdash \text{d2} : \text{Right\_recursive}}\ \text{(recurse-compile)}$$

0.4 Ty

The Ty is a datatype that contains all the information about the type itself. It contains the TyCons of the diagram, the source language, target language and the platform. They are contained in Maybes, because not all diagrams contain all information

$$\frac{}{\Gamma \Vdash \text{Program p l} : \text{(Ty Prog (just l) Nothing Nothing)}}\ \text{(ty-prog)}$$

$$\frac{}{\Gamma \Vdash \text{Interpreter i l m} : \text{(Ty Interp (Just l) Nothing (Just m))}}\ \text{(ty-interp)}$$

$$\frac{}{\Gamma \Vdash \text{Compiler c l1 l2 m} : \text{(Ty Comp (Just l1) (Just l2) (Just m))}}\ \text{(ty-comp)}$$

$$\frac{}{\Gamma \Vdash \text{Platform m} : \text{(Ty PlatF Nothing Nothing (Just m))}}\ \text{(ty-platf)}$$

$$\frac{\Gamma \Vdash \text{Execute d1} : \text{Ty d2} : \text{Ty}}{\Gamma \Vdash \text{d2} : \text{Ty}}\ \text{(ty-execute)}$$

$$\frac{\Gamma \Vdash \text{Compile d1} : \text{(Ty Prog s1 t1 m1) d2} : \text{(Ty Comp s2 t2 m2)}}{\Gamma \Vdash \text{(Ty Prog t2 t1 m1)} : \text{Ty}}\ \text{(ty-compile-prog)}$$

$$\frac{\Gamma \Vdash \text{Compile d1} : \text{(Ty Interp s1 t1 m1) d2} : \text{(Ty Comp s2 t2 m2)}}{\Gamma \Vdash \text{(Ty Interp s1 t1 t2)} : \text{Ty}}\ \text{(ty-compile-interp)}$$

$$\frac{\Gamma \Vdash \text{Compile d1} : \text{(Ty Comp s1 t1 m1) d2} : \text{(Ty Comp s2 t2 m2)}}{\Gamma \Vdash \text{(Ty Comp s1 t1 t2)} : \text{Ty}}\ \text{(ty-compile-comp)}$$

0.5 checkRunnable

CheckRunnable checks whether a platform is being executed or compiled. In otherwords, if the first argument is a PlatF, then it is ill-typed.

$$\frac{\Gamma \Vdash \text{t1} : \text{PlatF} \ \text{t2} : \text{TyCons}}{\Gamma \Vdash \text{ill-typed}} \ \scriptstyle(\text{checkrunnable-False})$$

0.6 checkComp

The checkComp is used to only accept a compiler when its type is Comp, because a Compile needs a compiler to compile.

$$\frac{\Gamma \Vdash \text{t1} : \text{TyCons} \ \text{t2} : \neg\text{Comp}}{\Gamma \Vdash \text{ill-typed}} \ \scriptstyle(\text{checkComp-False})$$

0.7 checkExeInComp

The checkExeInComp is used to refuse when a executed is in the compiler.

$$\frac{\Gamma \Vdash \text{t1} : \text{Executed} \bigvee \text{t2} : \text{Executed}}{\Gamma \Vdash \text{ill-typed}} \ \scriptstyle(\text{checkExeInComp-True})$$

0.8 checkExeOrComp

The checkExeOrComp is used to refuse an execution on a compilation or another execution

$$\frac{\Gamma \Vdash \text{t1} : \text{TyCons} \ (\text{t2} : \text{Executed} \bigvee \text{t2} : \text{Compiled})}{\Gamma \Vdash \text{ill-typed}} \ \scriptstyle(\text{checkExeOrComp-false})$$

0.9 checkFramework

The checkFramework is used to only accept an execution on a Interpreter or Platform

$$\frac{\Gamma \Vdash \text{t1} : \text{TyCons} \ \text{t2} : \neg\text{Framework}}{\Gamma \Vdash \text{ill-typed}} \ \scriptstyle(\text{checkFramework-False})$$

0.10 Checkifmatch

In the checkmatch we check on each type combination if it's being executed, compiled or interpreted on the same language as the source, target or platform. We use the matchInfo for this.

$$\frac{\Gamma \Vdash \text{t1 : (Ty Prog s1 t1 m1) t2 : (Ty Interp } \neg\text{s1 t2 m2)}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checkifmatch-prog-interp-ill)}$$

$$\frac{\Gamma \Vdash \text{t1: (Ty c s1 t1 m1) t2 : (Ty Interp } \neg\text{m1 t2 m2)}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checkifmatch-unknown-interp-ill)}$$

$$\frac{\Gamma \Vdash \text{t1 : (Ty Prog s1 t1 m1) t2 : (Ty PlatF s2 t2 } \neg\text{s1)}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checkifmatch-prog-platf-ill)}$$

$$\frac{\Gamma \Vdash \text{t1 : (Ty c s1 t1 m1) t2 : (Ty PlatF s2 t2 } \neg\text{m1)}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checkifmatch-unknown-platf-ill)}$$

$$\frac{\Gamma \Vdash \text{t1 : (Ty Prog s1 t1 m1) t2 : (Ty Comp } \neg\text{s1 t2 m2)}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checkifmatch-prog-comp-ill)}$$

$$\frac{\Gamma \Vdash \text{t1 : (Ty c s1 t1 m2) t2 : (Ty comp } \neg\text{m1 t2 m2)}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checkifmatch-unkown-comp-ill)}$$

0.11 checkLandRrecurs

Both subdiagrams of a compile cannot be Right_recursive and Left_recursive at the same time.

$$\frac{\Gamma \Vdash \text{t1 : Left\_recursive t2 : Right\_recursive}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checklandr)}$$

0.12 checkLeftRightRecurs

Diagram d1 of a compile is not allowed to have a left recursive compilation within. Likewise a diagram d2 of a compile is not allowed to have a right recursive compilation within.

Variable t is the recursion of the parent of t1

$$\frac{\Gamma \Vdash \text{t : Left\_recursive t1 : Right\_recursive}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checklr)}$$

$$\frac{\Gamma \Vdash \mathrm{t} : \mathrm{Right\_recursive}\ \mathrm{t1} : \mathrm{Left\_recursive}}{\Gamma \Vdash \text{ill-typed}}\ \text{(checklr)}$$