

## Formal typesystem description

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

November 30, 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.

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

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

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

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

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

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

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

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

Some TyCons also have a TyCons type

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

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

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

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

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

The following proofs determine the TyCons for each diagram

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

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

$$\frac{}{\Gamma \Vdash \text{Compiler } c \text{ l1 l2 m} : \text{Comp}} \quad (\text{tyCons-comp})$$

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

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

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

### 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.

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

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

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

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

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

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

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

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

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

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

## 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

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

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

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

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

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

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

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

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

## 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.

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

## 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.

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

## 0.7 checkExeInComp

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

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

## 0.8 checkExeOrComp

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

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

## 0.9 checkFramework

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

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

## 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.

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

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

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

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

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

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

## 0.11 checkLandRrecurs

Both subdiagrams of a compile cannot be Right\_recursive and Left\_recursive at the same time.

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

## 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

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

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