*Fundamenta Informaticae 128 (2013) 17–33*
*DOI 10.3233/FI-2013-930*
*IOS Press*

# Java SAM Typed Closures: A Sound and Complete Type Inference System for Nominal Types

**Marco Bellia and M. Eugenia Occhiuto**[*]
*Dipartimento di Informatica, Università di Pisa, Italy*
*{bellia,occhiuto}@di.unipi.it*

**Abstract.** The last proposal for Java closures, as emerged in JSR 000335, is mainly innovative in: (1)Use of nominal types, SAM types, for closures; (2) Introduction of target types and compatibility for a contextual typing of closures; (3) Need for a type inference that reconstructs the omitted type annotations of closures and closure arguments. The paper provides a sound and complete type system, with nominal types, for such a type inference and discusses role and formalization of targeting and of compatibility in the designed inference process.

## 1. Introduction

The paper provides a type inference system, with nominal types, for closures that are typed with SAM types (hence, the name of SAM typed closures), that: 1) it is sound and complete; 2) given a program, it checks for the existence of an assignment of types to the omitted type annotations that make the resulting program, correctly typed; 3) if an assignment exists, it results in the most general assignment of the program. 4) it works with nominal types. Nominal types have both an external structure, i.e. the name of the type, and an internal structure, i.e. the type expression defining form and components of the type. Hence, type inference with nominal types: *i)* must provide the usual mechanism that reconstructs the (internal) structure that the omitted type annotations must have; *ii)* requires an additional mechanism to select, among all the nominal types whose internal structure matches the found one, the type that must be considered the most general correct type.

JSR 000335 [6, 7] shares with the other previous, proposals [2, 13] the idea of introducing closures as expressions defining shortenings for anonymous, single method, objects, but it is innovative in many fundamental aspects.

[*]Address for correspondence: Dipartimento di Informatica, Università di Pisa, Italy

**Closure Definition.** Closures are introduced, in a program, by a special form of expressions (*lambda expressions*). The syntax of a closure definition consists of the the generic types (if any), the argument name list (possibly, empty), and the closure body.

**Type Inference.** The closure definition does not require a type annotation [14] of the defined closure and also, the type annotations of the arguments can be omitted: Strong typing and omitted type annotations require a type system capable to perform type inference. A failure during the inference process, causes a type failure.

**SAM Types.** Interface types with a single method, named *functional interfaces* (*SAM* types [1]), are the types of the closures. As all Java reference types, SAM types are nominal types, i.e they are different types if they have different names even though they have the same internal structure. For this reason, a closure can be assigned to (i.e. is *compatible* with) many different types.

**Generic Types.** Generics may be introduced in a closure definition and must be the same, after renaming and permutation, of the generics that have been introduced in the closure SAM type. A problem arises: How can such a SAM type be found when the closure type annotation was omitted?

**Target Types.** The solution adopted is to assign to each closure the *target* type, that is the expected type in the specific *context* in which the closure is used. The target type becomes the type if it is compatible with the closure.

**Closure Contexts.** The possible contexts in which a closure can appear are:

1. Variable declaration
2. Assignment
3. Return statement
4. Array initializer
5. Method or constructor argument
6. Lambda expression body
7. Conditional expression
8. Cast expression

**Type Compatibility.** The conditions which must hold for a closure to be compatible with a type are: *i)* The type must be a functional interface: Let *m* be its single method. *ii)* Number and types of the closure arguments must be the same as those of *m*. *iii)* Return types of the closure and of *m* must be compatible. *iv)* Exceptions thrown by the closure body must be allowed in the `throws` clause of *m*.

**Closure Invocation.** There is no ad hoc syntax for closure invocation. The user has to specify, hence know and remember, the name that has been chosen for the single method of the functional interface.

**Non-local Variables.** Any name used but not declared in the closure, must be either declared `final` or *effectively final*. The concept of *effectively final* variables, already introduced in Java SE 7, is now broadened, to mitigate the restriction on variables updating. An effectively final variable is a variable which is not declared final but its value is not modified.

**Variable Shadowing.** As for blocks the local variables or formal parameters of a closure cannot shadow already declared names.

**Meaning of `this`.** The self reference `this` in a closure refers to the object whose method is enclosing the closure definition and not to the defined closure (`this` transparency), thus disallowing recursive definitions through `this`. To define a recursive closure, it's necessary to associate a name to the closure, for instance through variable declaration and initialization or assignment.

For space reason we limit the definition of the type inference system to a significative kernel of Java with closures: It includes the field initialization but leaves out variables and assignment, includes interfaces and hierarchies of classes but leaves out hierarchies of interfaces, includes method overriding but leaves out method overloading.

We assume the reader already has some familiarity with the algebraic framework of *Featherweight Java*, introduced in [11], in particular with its main technicalities and motivations for their use. Then, the paper organization is as follows: We start defining a reduction semantics and declarative typing system for a kernel of Java, FGJ [11], extended with SAM typed closures, FGATCJ. The term ”declarative” is in order to emphasize that the closures that we initially consider, in FGATCJ, are SAM typed closures that have type annotations of the defined closure and of all the closure arguments. The typing system is then, studied to provide soundness and to formalize various notions of the proposal, including the type targeting and type compatibility. Then, we extend FGATCJ in FGATCJ$^\bullet$ that differs because of its type structure and of its typing system. The type structure includes a (denumerable) set of a new kind of variables, called d-variables. Programs of FGATCJ$^\bullet$ are considered as the counterpart of programs of FGATCJ where some type annotations, in the program closure definitions, are omitted and replaced by d-variables. The typing system of FGATCJ$^\bullet$ is in effect, a type inference system: The system introduces constrained judgements that are essentially the judgements of the declarative typing system, extended with the constraints of a suitable constraint system. A constrain solver, based on ordinary, first order unification, is then, defined: It computes the most general solution. We prove that the inference system is sound and complete, and we show how, given a program, it computes the most general assignment of d-variable to types, that makes the resulting program correctly typed, provided that one such an assignment exists.

Section 2 contains a brief presentation of the Java kernel language FGJ. Section 3 introduces the kernel language FGATCJ and its reduction semantics. Section 4 contains the (declarative) typing system of FGATCJ and proves the type soundness. Section 5 introduces the kernel language FGATCJ$^\bullet$, the type inference system and proves soundness and completeness. Section 6 concludes the paper.

## 2. Featherweight Generic Java

A program in FGJ [11] consists of a collection of generic class definitions and of an expression to be evaluated using such classes. The expression corresponds to the body of the 0-arguments main method of ordinary Java.

A complete definition of the abstract syntax of FGJ consists of the grammar rules in **Table 1** that are labelled by the defined grammatical category indexed by FGJ. Symbols $\triangleleft$ and $\uparrow$ are a notational shorthands for Java keyword `extends` and `return`. For syntactic regularity, (a) classes always specify the super class, possibly `Object`, and have exactly one constructor definition; (b) class constructors have one parameter for each class field with the same name as the field, invoke the super constructor on the fields of the super class and initialize the remaining fields to the corresponding parameters; (c) field access always specifies the receiver (object), possibly `this`. This results in the stylized form of the constructors. Both classes and methods may have generic type parameters.

FGJ has no side effects. Hence, *sequencing* and *assignment* are strictly confined to constructor bodies. In particular, method bodies have always the form `return`, followed by an expression. The lack of Java constructs for sequencing control and for store updating (along with that of concurrency, and reflection) is the main advantage of the calculus in studying language properties that are not affected by side effects. In this way the calculus is, as much as possible, compact and takes advantage of the referential transparency. The latter one provides a simple reduction semantics which is crucial for rigorous, easy to derive, proofs of the language properties [10]. About compactness, FGJ has only five forms of expressions (see definition of category `e` in **Table 1**): One for variables[1], another for *field access*, and one for *Object Creation*. The remaining two forms are *method invocation* and *cast*.

The presence of *cast* in FGJ is justified by its fundamental role in compiling generic classes. The reduction semantics of FGJ consists of the first three rules that appear in **Table 2: Computation**, and deal with term evaluation, and of the first five rules in **Table 2: Congruence**, that deal with the redex selection. The remaining 20 rules of the semantics of FGJ deal with the type system and with term well-formedness. The rules of FGJ have labels that are indexed by FGJ in **Tables 2, 4, 5**.

## 3. Featherweight GATCJ

The calculus defined in this paper is obtained as an extension of the calculus FGAJ[3], which is in turn an extension of FGJ, with interfaces, anonymous classes and consequently objects from anonymous classes creation. FGATCJ extends FGAJ with the closures defined in [6]. The issue concerned with non-local variables in closures, has no meaning since all the variables of FGJ can be considered effectively final. Similarly, it is about variable shadowing, since FGJ programs are, in effect, abstract syntax terms (i.e. modulo variable renaming).

### 3.1. Notation and General Conventions

In this paper we adopt the notation used in [11], accordingly $\overline{\mathtt{f}}$ is a shorthand for a possibly empty sequence $\mathtt{f}_1, \ldots, \mathtt{f}_n$ (and similarly for $\overline{\mathtt{T}}, \overline{\mathtt{x}}$, etc.) and $\overline{\mathtt{M}}$ is a shorthand for $\mathtt{M}_1 \ldots \mathtt{M}_n$ (with no commas) where $n$ is the size $|\overline{\mathtt{f}}|$, respectively $|\overline{\mathtt{M}}|$, i.e. the number of terms of the sequence. The empty sequence is $\circ$ and symbol "," denotes concatenation of sequences. Operations on pairs of sequences are abbreviated in the obvious way $\overline{\mathtt{C}}\ \overline{\mathtt{f}}$ is $\mathtt{C}_1\ \mathtt{f}_1, \ldots, \mathtt{C}_n\ \mathtt{f}_n$ and similarly $\overline{\mathtt{C}}\ \overline{\mathtt{f}};$ is $\mathtt{C}_1\ \mathtt{f}_1; \ldots \mathtt{C}_n\ \mathtt{f}_n;$ and $\mathtt{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};$ is a shorthand for $\mathtt{this}.\mathtt{f}_1 = \mathtt{f}_1; \ldots \mathtt{this}.\mathtt{f}_n = \mathtt{f}_n;$. Sequences of field declarations, parameters and method declaration cannot contain duplications. Cast, $(\_)\_$, and closure definition, $\_ \to \_$, have lower precedence than other operators, and cast precedes closure definition. Hence $() \to (\mathtt{this.invoke}())$ can be written as $() \to \mathtt{this.invoke}()$. The, possibly indexed and/or primed, metavariables `T`, `V`, `U`, `S`, `W` range over type expressions; `X`, `Y`, `Z` range over type variables; `N`, `P`, `Q` range over class types; `C`, `D`, `E` range over class names; `I` ranges over interface names; `f`, `g` range over field names; `e`, `v`, `d` range over expressions; `x`, `y` range over variable names and `M`, `K`, `L`, `H` and `m` range respectively, over methods, constructors, classes

[1]Variables include parameters and `this`, see rule `GR-Invk`, **Table 2**.

and interfaces, method headers, and method names. $[x/y]e$ denotes the result of replacing $y$ by $x$ in $e$. $FV(\overline{\mathtt{T}})$ denotes the set of free type variables in $\overline{\mathtt{T}}$. Eventually, following the notation adopted in [11], symbol "=" is used in formulas both as an abbreviation for *let* $\Delta = \overline{\mathtt{X}}{<:}\overline{\mathtt{N}}, \dots$ and as a constraint *if* $\overline{\mathtt{X}} = FV(\overline{\mathtt{T}})$: The context solves any ambiguity.

## 3.2. Syntax

A program in FGATCJ consists of a collection of generic, class and interface definitions and of an expression to be evaluated using such classes and interfaces. The syntax is formally, given in **Table 1** by emphasizing the extensions of which FGATCJ is composed starting from the kernel language: At the top of the table is reported FGJ, below the first extension IA (that involves extensions on both types and expressions), finally the last extension TC that introduces the closures. The table concludes with a view of the various involved, languages.

Besides interfaces and anonymous classes already provided in FGAJ and FGACJ in [3], in FGATCJ it is possible to define SAM typed closures. The syntax of the expressions that define SAM closures is given in **Table 1**. Such expressions have form $\mathtt{a} : \mathtt{T}$, where: $\mathtt{T}$ is the type annotation of the defined closure and specifies the type that the defined closure is supposed to have. Moreover, the form of $\mathtt{a}$ is $\langle \overline{\mathtt{X}} \lhd \overline{\mathtt{N}} \rangle (\overline{\mathtt{T}}\ \overline{\mathtt{x}}) \to \mathtt{e}$ where: $\langle \overline{\mathtt{X}} \lhd \overline{\mathtt{N}} \rangle$ is the possibly empty, sequence of the closure generics, whilst $\mathtt{e}$ is the closure body. Eventually, $\overline{\mathtt{T}}\ \overline{\mathtt{x}}$ is the possibly empty, argument sequence of the closure, where: each argument $\mathtt{x}_i$ has associated its type annotation $\mathtt{T}_i$ that specifies the type that the defined closure assumes for the argument $\mathtt{x}_i$.

**Table 1 : Syntax**

| | |
|---|---|
| FGJ | |
| $\mathtt{T} ::= \mathtt{X} \mid \mathtt{N}$ | $(\mathtt{T}_{\mathrm{FGJ}})$ |
| $\mathtt{N} ::= \mathtt{C}\langle \overline{\mathtt{T}} \rangle$ | $(\mathtt{N}_{\mathrm{FGJ}})$ |
| $\mathtt{L} ::= \mathtt{class}\ \mathtt{C}\langle \overline{\mathtt{X}} \lhd \overline{\mathtt{N}} \rangle \lhd \mathtt{N}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}}; \mathtt{K}\ \overline{\mathtt{M}}\}$ | $(\mathtt{L}_{\mathrm{FGJ}})$ |
| $\mathtt{K} ::= \mathtt{C}(\overline{\mathtt{T}}\ \overline{\mathtt{f}})\{\mathtt{super}(\overline{\mathtt{f}}); \mathtt{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};\}$ | $(\mathtt{K}_{\mathrm{FGJ}})$ |
| $\mathtt{M} ::= \langle \overline{\mathtt{X}} \lhd \overline{\mathtt{N}} \rangle \mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\ \overline{\mathtt{x}})\{\uparrow \mathtt{e};\}$ | $(\mathtt{M}_{\mathrm{FGJ}})$ |
| $\mathtt{e} ::= \mathtt{x} \mid \mathtt{e.f} \mid \mathtt{new}\ \mathtt{N}(\overline{\mathtt{e}}) \mid \mathtt{e.m}\langle \overline{\mathtt{T}} \rangle(\overline{\mathtt{e}}) \mid (\mathtt{N})\mathtt{e}$ | $(\mathtt{e}_{\mathrm{FGJ}})$ |
| IA: *Extensions for Interfaces and Anonymous Class Objects* | |
| $\mathtt{T} ::= \mathtt{I}\langle \overline{\mathtt{T}} \rangle$ | $(\mathtt{T}_{\mathrm{FGAJ}})$ |
| $\mathtt{L} ::= \mathtt{interface}\ \mathtt{I}\langle \overline{\mathtt{X}} \lhd \overline{\mathtt{N}} \rangle\{\overline{\mathtt{H}}\}$ | $(\mathtt{L}_{\mathrm{FGAJ}})$ |
| $\mathtt{H} ::= \langle \overline{\mathtt{X}} \lhd \overline{\mathtt{N}} \rangle \mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\ \overline{\mathtt{x}})$ | $(\mathtt{H}_{\mathrm{FGAJ}})$ |
| $\mathtt{e} ::= \mathtt{new}\ \mathtt{I}\langle \overline{\mathtt{T}} \rangle()\ \{\overline{\mathtt{M}}\}$ | $(\mathtt{e}_{\mathrm{FGAJ}})$ |
| TC: *Extensions for SAM Typed Closures* | |
| $\mathtt{e} ::= \mathtt{a} : \mathtt{T} \mid (\mathtt{I}\langle \overline{\mathtt{T}} \rangle)\mathtt{e}$ | $(\mathtt{e}_{\mathrm{FGATCJ}})$ |
| $\mathtt{a} ::= \langle \overline{\mathtt{X}} \lhd \overline{\mathtt{N}} \rangle(\overline{\mathtt{T}}\ \overline{\mathtt{x}}) \to \mathtt{e}$ | $(\mathtt{a}_{\mathrm{FGATCJ}})$ |
| FGAJ = FGJ + IA | |
| FGATCJ = FGJ + IA + TC | |

Actually, these closures differ from the ones in [6] because they always require type annotations for both the defined closure and the closure arguments. In effect, the aim of FGATCJ is to provide the right (language) structure to design a type system for SAM typed closure, study the nominal properties of SAM types and prove the soundness of the system. In this respect, the types of the defined closures as well as those of closure arguments are explicitly given and the proposed type system is designed only for checking the type correctness of the programs. The type system of FGATCJ is a sort of declarative type system for a language, later on called FGATCJ$^{\bullet}$, in which type annotations can be omitted as in the closures of [6], hence the name used in Section 4 and in **Table 4DN**: In particular, the programs of FGATCJ are those of FGATCJ$^\bullet$ where the omitted type annotations are replaced with types and the resulting programs are then checked for type correctness.

The possibility to omit types in FGATCJ$^\bullet$ is dealt with in Section 5 where, the type correctness of programs of FGATCJ$^\bullet$ becomes the existence of a suitable type assignment for the omitted type annotations, whilst checking for type correctness leads to design a type inference for asking about such a type assignment. The type inference will be obtained from the declarative type system through a suitable re-design of the judgments introduced in Section 5.1.

For space convenience, the reduction rules of the semantics as well as the typing rules are not given in separate tables for each calculus. In fact, since compositionality of the semantics (we use), the rules of the various constructs are the same in all calculi containing such a construct. However, for the reader convenience, in all tables, but **Table 3**, the rules for each calculus, FGJ, FGAJ and FGATCJ, have a label which is indexed by the name of the minimal calculus including the construct, involved in the rule. Note that $\mathtt{C}\langle\overline{\mathtt{T}}\rangle$ includes `Object`(since $\overline{\mathtt{T}}$ may be the empty sequence and `C` may be `Object`) hence generic variables in classes and methods can be instantiated with types `T` that include interfaces.

## 3.3. Semantics: Reduction

**Table 2: Computation**

| | |
|---|---|
| **Computation** | |
| $\dfrac{\textit{fields}(\mathtt{N}) = \overline{\mathtt{T}}\,\overline{\mathtt{f}}}{(\mathtt{new\ N}(\overline{\mathtt{e}})).\mathtt{f}_i \rightsquigarrow \mathtt{e}_i}$ | (GR-FIELD$_{\text{FGJ}}$) |
| $\dfrac{\textit{mbody}(\mathtt{m}\langle\overline{\mathtt{V}}\rangle, \mathtt{N}) = \overline{\mathtt{x}}.\mathtt{e}}{(\mathtt{new\ N}(\overline{\mathtt{e}})).\mathtt{m}\langle\overline{\mathtt{V}}\rangle(\overline{\mathtt{d}}) \rightsquigarrow [\overline{\mathtt{d}}/\overline{\mathtt{x}}, \mathtt{new\ N}(\overline{\mathtt{e}})/\mathtt{this}]\mathtt{e}}$ | (GR-INVK$_{\text{FGJ}}$) |
| $\dfrac{\emptyset \vdash \mathtt{N} <: \mathtt{P}}{(\mathtt{P})(\mathtt{new\ N}(\overline{\mathtt{e}})) \rightsquigarrow \mathtt{new\ N}(\overline{\mathtt{e}})}$ | (GR-CAST$_{\text{FGJ}}$) |
| $\dfrac{\textit{mbody}(\mathtt{m}\langle\overline{\mathtt{V}}\rangle, \mathtt{new\ I}\langle\overline{\mathtt{T}}\rangle()\{\overline{\mathtt{M}}\}) = \overline{\mathtt{x}}.\mathtt{e}}{(\mathtt{new\ I}\langle\overline{\mathtt{T}}\rangle()\{\overline{\mathtt{M}}\}).\mathtt{m}\langle\overline{\mathtt{V}}\rangle(\overline{\mathtt{d}}) \rightsquigarrow [\overline{\mathtt{d}}/\overline{\mathtt{x}}, \mathtt{new\ I}\langle\overline{\mathtt{T}}\rangle()\{\overline{\mathtt{M}}\}/\mathtt{this}]\mathtt{e}}$ | (GR-INVK-ANONYM$_{\text{FGAJ}}$) |
| $(\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle(\overline{\mathtt{T}}\,\overline{\mathtt{x}}) \rightarrow \mathtt{e} : \mathtt{T}).\mathtt{m}\langle\overline{\mathtt{S}}\rangle(\overline{\mathtt{d}}) \rightsquigarrow [\overline{\mathtt{d}}/\overline{\mathtt{x}}]\mathtt{e}$ | (GR-CLOS-INV-TYPE$_{\text{FGATCJ}}$) |
| $(\mathtt{T})\,\mathtt{a} : \mathtt{T} \rightsquigarrow \mathtt{a} : \mathtt{T}$ | (GR-CCAST$_{\text{FGATCJ}}$) |
| **Congruence** | |
| $\dfrac{\mathtt{e}_0 \rightsquigarrow \mathtt{e}_0'}{\mathtt{e}_0.\mathtt{f} \rightsquigarrow \mathtt{e}_0'.\mathtt{f}}$ | (GRC-FIELD$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{e}_0 \rightsquigarrow \mathtt{e}_0'}{\mathtt{e}_0.\mathtt{m}\langle\overline{\mathtt{T}}\rangle(\overline{\mathtt{e}}) \rightsquigarrow \mathtt{e}_0'.\mathtt{m}\langle\overline{\mathtt{T}}\rangle(\overline{\mathtt{e}})}$ | (GRC-T-INV$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{e}_i \rightsquigarrow \mathtt{e}_i'}{\mathtt{e}_0.\mathtt{m}\langle\overline{\mathtt{T}}\rangle(\ldots, \mathtt{e}_i, \ldots) \rightsquigarrow \mathtt{e}_0.\mathtt{m}\langle\overline{\mathtt{T}}\rangle(\ldots, \mathtt{e}_i' \ldots)}$ | (GRC-INV-ARG$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{e}_i \rightsquigarrow \mathtt{e}_i'}{\mathtt{new\ N}(\ldots, \mathtt{e}_i, \ldots) \rightsquigarrow \mathtt{new\ N}(\ldots, \mathtt{e}_i', \ldots)}$ | (GRC-NEW$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{e} \rightsquigarrow \mathtt{e}'}{(\mathtt{N})\mathtt{e} \rightsquigarrow (\mathtt{N})\mathtt{e}'}$ | (GRC-CAST$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{e} \rightsquigarrow \mathtt{e}'}{(\mathtt{I}\langle\overline{\mathtt{T}}\rangle)\mathtt{e} \rightsquigarrow (\mathtt{I}\langle\overline{\mathtt{T}}\rangle)\mathtt{e}'}$ | (GRC-CCAST$_{\text{FGATCJ}}$) |

The reduction semantics is given through the inference rules in **Table 2**, which define the relation $\mathtt{e} \rightsquigarrow \mathtt{e}'$ that says that "expression $\mathtt{e}$ reduces to expression $\mathtt{e}'$ in one step". The computation may diverge or otherwise, terminate, possibly resulting into a value. The values of FGATCJ are either named or anonymous class objects and closures. The syntactic category $\mathcal{V}$ defines the form of the expressions representing such values:

$$\begin{array}{rcl} \mathcal{V} & ::= & \mathtt{new}\,\mathtt{N}(\overline{\mathcal{V}}) \\ & & \mid \mathtt{new}\,\mathtt{I}\langle\overline{\mathtt{T}}\rangle()\{\overline{\mathtt{M}}\} \\ & & \mid \langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle(\overline{\mathtt{T}}\,\overline{\mathtt{x}}) \rightarrow \mathtt{e} : \mathtt{T} \end{array}$$

These expressions are normal forms (i.e. cannot be further reduced) of the reduction relation $\rightsquigarrow$, as it was expected. However, the converse does not hold since expressions as they result from the grammar in **Table 1** may contain unproper uses of field accesses, or of method invocations, or of type casts: The type system discussed in the next sections, aims to recognize programs that contain such expressions.

**Table 3: Classes and Interfaces**

**Subclassing**

$$\mathtt{C} \trianglelefteq \mathtt{C} \qquad \frac{\mathtt{C} \trianglelefteq \mathtt{D} \qquad \mathtt{D} \trianglelefteq \mathtt{E}}{\mathtt{C} \trianglelefteq \mathtt{E}} \qquad \frac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{D}\ \{\overline{\mathtt{S}}\,\overline{\mathtt{f}};\mathtt{K}\,\overline{\mathtt{M}}\}}{\mathtt{C} \trianglelefteq \mathtt{D}}$$

**Auxiliary functions**

$$\mathit{fields}(\mathtt{Object}) = \circ \qquad \text{(F-OBJECT)}$$

$$\frac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\ \{\overline{\mathtt{S}}\,\overline{\mathtt{f}};\mathtt{K}\,\overline{\mathtt{M}}\} \qquad \mathit{fields}([\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{N}) = \overline{\mathtt{U}}\,\overline{\mathtt{g}}}{\mathit{fields}(\mathtt{C}\langle\overline{\mathtt{T}}\rangle) = \overline{\mathtt{U}}\,\overline{\mathtt{g}},\ [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{S}}\,\overline{\mathtt{f}}} \qquad \text{(F-CLASS)}$$

$$\frac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\ \{\overline{\mathtt{S}}\,\overline{\mathtt{f}};\mathtt{K}\,\overline{\mathtt{M}}\} \qquad \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{U}\,\mathtt{m}\,(\overline{\mathtt{U}}\,\overline{\mathtt{x}})\{\uparrow \mathtt{e};\} \in \overline{\mathtt{M}}}{\mathit{mtype}(\mathtt{m}, \mathtt{C}\langle\overline{\mathtt{T}}\rangle) = [\overline{\mathtt{T}}/\overline{\mathtt{X}}](\langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\overline{\mathtt{U}} \mapsto \mathtt{U})} \qquad \text{(MT-CLASS)}$$

$$\frac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\ \{\overline{\mathtt{S}}\,\overline{\mathtt{f}};\mathtt{K}\,\overline{\mathtt{M}}\} \qquad \mathtt{m} \not\in \overline{\mathtt{M}}}{\mathit{mtype}(\mathtt{m}, \mathtt{C}\langle\overline{\mathtt{T}}\rangle) = \mathit{mtype}(\mathtt{m}, [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{N})} \qquad \text{(MT-SUPER)}$$

$$\frac{\mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\ \{\overline{\mathtt{H}}\} \qquad \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{U}\,\mathtt{m}(\overline{\mathtt{U}}\,\overline{\mathtt{x}}) \in \overline{\mathtt{H}}}{\mathit{mtype}(\mathtt{m}, \mathtt{I}\langle\overline{\mathtt{T}}\rangle) = [\overline{\mathtt{T}}/\overline{\mathtt{X}}](\langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\overline{\mathtt{U}} \mapsto \mathtt{U})} \qquad \text{(MT-INTERFACE)}$$

$$\frac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\ \{\overline{\mathtt{S}}\,\overline{\mathtt{f}};\mathtt{K}\,\overline{\mathtt{M}}\} \qquad \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{U}\,\mathtt{m}\,(\overline{\mathtt{U}}\,\overline{\mathtt{x}})\{\uparrow \mathtt{e};\} \in \overline{\mathtt{M}}}{\mathit{mbody}(\mathtt{m}\langle\overline{\mathtt{V}}\rangle, \mathtt{C}\langle\overline{\mathtt{T}}\rangle) = \overline{\mathtt{x}}.[\overline{\mathtt{T}}/\overline{\mathtt{X}}, \overline{\mathtt{V}}/\overline{\mathtt{Y}}]\mathtt{e}} \qquad \text{(MB-CLASS)}$$

$$\frac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\ \{\overline{\mathtt{S}}\,\overline{\mathtt{f}};\mathtt{K}\,\overline{\mathtt{M}}\} \qquad \mathtt{m} \not\in \overline{\mathtt{M}}}{\mathit{mbody}(\mathtt{m}\langle\overline{\mathtt{V}}\rangle, \mathtt{C}\langle\overline{\mathtt{T}}\rangle) = \mathit{mbody}(\mathtt{m}\langle\overline{\mathtt{V}}\rangle, [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{N})} \qquad \text{(MB-SUPER)}$$

$$\frac{\mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\ \{\ldots\} \qquad \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{U}\,\mathtt{m}\,(\overline{\mathtt{U}}\,\overline{\mathtt{x}})\{\uparrow \mathtt{e};\} \in \overline{\mathtt{M}}}{\mathit{mbody}(\mathtt{m}\langle\overline{\mathtt{V}}\rangle, \mathtt{new}\ \mathtt{I}\langle\overline{\mathtt{T}}\rangle()\{\overline{\mathtt{M}}\}) = \overline{\mathtt{x}}.[\overline{\mathtt{T}}/\overline{\mathtt{X}}, \overline{\mathtt{V}}/\overline{\mathtt{Y}}]\mathtt{e}} \qquad \text{(MB-INTERFACE)}$$

$$\frac{\mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\ \{\overline{\mathtt{H}}\} \qquad |\overline{\mathtt{H}}| = 1 \qquad \Delta \vdash \overline{\mathtt{V}} <: [\overline{\mathtt{V}}/\overline{\mathtt{X}}]\overline{\mathtt{N}} \qquad \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{U}\,\mathtt{m}\,(\overline{\mathtt{U}}\,\overline{\mathtt{x}}) = \overline{\mathtt{H}}}{\Delta \vdash \mathit{met}(\mathtt{I}\langle\overline{\mathtt{V}}\rangle) = \langle\overline{\mathtt{Y}} \lhd [\overline{\mathtt{V}}/\overline{\mathtt{X}}]\overline{\mathtt{P}}\rangle\,[\overline{\mathtt{V}}/\overline{\mathtt{X}}]\mathtt{U}\,\mathtt{m}\,([\overline{\mathtt{V}}/\overline{\mathtt{X}}]\overline{\mathtt{U}}\,\overline{\mathtt{x}})} \qquad \text{(METHOD)}$$

**Auxiliary predicates**

$$\mathtt{override}(\mathtt{m}, \mathtt{Object}, \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\overline{\mathtt{T}} \mapsto \mathtt{T}_0) \qquad \text{(OVER-Object)}$$

$$\frac{\begin{array}{l}\mathit{mtype}(\mathtt{m}, \mathtt{N}) = \langle\overline{\mathtt{Z}} \lhd \overline{\mathtt{Q}}\rangle\overline{\mathtt{U}} \mapsto \mathtt{U}_0 \implies \\ \qquad ((\overline{\mathtt{P}}, \overline{\mathtt{T}}) = [\overline{\mathtt{Y}}/\overline{\mathtt{Z}}](\overline{\mathtt{Q}}, \overline{\mathtt{U}})\ \mathtt{and}\ \overline{\mathtt{Y}} <: \overline{\mathtt{P}} \vdash \mathtt{T}_0 <: [\overline{\mathtt{Y}}/\overline{\mathtt{Z}}]\mathtt{U}_0)\end{array}}{\mathtt{override}(\mathtt{m}, \mathtt{N}, \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\overline{\mathtt{T}} \mapsto \mathtt{T}_0)} \qquad \text{(OVER)}$$

$$\frac{\mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\ \{\overline{\mathtt{H}}\} \qquad \Delta \vdash \overline{\mathtt{V}} <: [\overline{\mathtt{V}}/\overline{\mathtt{X}}]\overline{\mathtt{N}} \qquad |\overline{\mathtt{H}}| = 1}{\Delta \vdash \mathtt{Fun}(\mathtt{I}\langle\overline{\mathtt{V}}\rangle)} \qquad \text{(FUN)}$$

**DCast**

$$\frac{dcast(\mathtt{C}, \mathtt{D}) \qquad dcast(\mathtt{D}, \mathtt{E})}{dcast(\mathtt{C}, \mathtt{E})} \qquad \frac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{D}\langle\overline{\mathtt{T}}\rangle\ \{\ldots\} \qquad \overline{\mathtt{X}} = FV(\overline{\mathtt{T}})}{dcast(\mathtt{C}, \mathtt{D})} \qquad \text{(DCAST)}$$

The structure of values results from the reduction rules of the calculus. The rules indexed by FGJ in **Table 2** are the same as those of calculus FGJ [11], and the one indexed by FGAJ is the same of FGAJ [3]. The rules use auxiliary functions ($mbody$, $fields$) and notation that are introduced in **Table 3**, which collects all the auxiliary definitions. The rule GR-INVK-ANONYM$_{\text{FGAJ}}$ defines the semantics of invocation with anonymous class objects, in a way quite similar to the one of method invocation with objects of named classes. The new rules indexed FGATCJ include invocation for closures and a rule to cast closures to SAM types. Also one congruence rule for casting is added.

## 4. Declarative Typing

The declarative typing extends the typing rules of [11], uses the same two environments $\Delta$ and $\Gamma$, and eight different typing judgements (two more than [11]'s ones): One judgment for each different term structure of the language. A type environment $\Delta$ is a mapping from type variables to types. It is written as a list of $\mathtt{X}{<:}\mathtt{T}$ (with at most one binding for each type variable $\mathtt{X}$), meaning that type variable $\mathtt{X}$ must be bound to a subtype of type $\mathtt{T}$. Hence, using functional notation, we have that $\Delta(\mathtt{X}) = \mathtt{T}$ holds if and only if $\Delta$ contains $\mathtt{X}{<:}\mathtt{T}$. An environment $\Gamma$ is a mapping from variables to types written as a list of $\mathtt{x} : \mathtt{T}$ (with at most one binding for each value variable $\mathtt{x}$), meaning that "$\mathtt{x}$ has type $\mathtt{T}$".

**Table 4DN: Declarative Typing Rules**

$$\Delta; \Gamma_1, \mathtt{x} : \mathtt{T}, \Gamma_2 \vdash \mathtt{x} : \mathtt{T} \qquad (\text{GT-VAR}_{\text{FGJ}})$$

$$\frac{\Delta; \Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \mathit{fields}(\mathit{bound}_\Delta(\mathtt{T}_0)) = \overline{\mathtt{T}}\,\overline{\mathtt{f}}}{\Delta; \Gamma \vdash \mathtt{e}_0.\mathtt{f}_i : \mathtt{T}_i} \qquad (\text{GT-FIELD}_{\text{FGJ}})$$

$$\frac{\begin{array}{c}\mathit{mtype}(\mathtt{m}, \mathit{bound}_\Delta(\mathtt{T}_0)) = \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\overline{\mathtt{U}} \mapsto \mathtt{U}\\ \Delta; \Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \Delta \vdash \overline{\mathtt{V}}\ \mathtt{ok} \qquad \Delta \vdash \overline{\mathtt{V}}{<:}[\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\overline{\mathtt{P}}\\ \Delta; \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{S}} \qquad \Delta \vdash \overline{\mathtt{S}}{<:}[\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\overline{\mathtt{U}}\end{array}}{\Delta; \Gamma \vdash \mathtt{e}_0.\mathtt{m}\langle\overline{\mathtt{V}}\rangle(\overline{\mathtt{e}}) : [\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\mathtt{U}} \qquad (\text{GT-INV}_{\text{FGJ}})$$

$$\frac{\begin{array}{c}\Delta \vdash \mathtt{N}\ \mathtt{ok} \qquad \mathit{fields}(\mathtt{N}) = \overline{\mathtt{T}}\,\overline{\mathtt{f}}\\ \Delta; \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{S}} \qquad \Delta \vdash \overline{\mathtt{S}}{<:}\overline{\mathtt{T}}\end{array}}{\Delta; \Gamma \vdash \mathtt{new}\ \mathtt{N}(\overline{\mathtt{e}}) : \mathtt{N}} \qquad (\text{GT-NEW}_{\text{FGJ}})$$

$$\frac{\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T} \qquad \Delta \vdash \mathit{bound}_\Delta(\mathtt{T}){<:}\mathtt{N}}{\Delta; \Gamma \vdash (\mathtt{N})\mathtt{e} : \mathtt{N}} \qquad (\text{GT-UCAST}_{\text{FGJ}})$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T} \qquad \Delta \vdash \mathtt{N}\ \mathtt{ok} \qquad \Delta \vdash \mathtt{N}{<:}\mathit{bound}_\Delta(\mathtt{T})\\ \mathtt{N} = \mathtt{C}\langle\overline{\mathtt{T}}\rangle \qquad \mathit{bound}_\Delta(\mathtt{T}) = \mathtt{D}\langle\overline{\mathtt{U}}\rangle \qquad \mathit{dcast}(\mathtt{C}, \mathtt{D})\end{array}}{\Delta; \Gamma \vdash (\mathtt{N})\mathtt{e} : \mathtt{N}} \qquad (\text{GT-DCAST}_{\text{FGJ}})$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T} \qquad \Delta \vdash \mathtt{N}\ \mathtt{ok}\\ \mathtt{N} = \mathtt{C}\langle\overline{\mathtt{T}}\rangle \qquad \mathit{bound}_\Delta(\mathtt{T}) = \mathtt{D}\langle\overline{\mathtt{U}}\rangle \qquad \mathtt{C} \not\trianglelefteq \mathtt{D} \qquad \mathtt{D} \not\trianglelefteq \mathtt{C}\end{array}}{\Delta; \Gamma \vdash (\mathtt{N})\mathtt{e} : \mathtt{N}} \qquad (\text{GT-SCAST}_{\text{FGJ}})$$

$$\frac{\Delta \vdash \mathtt{I}\langle\overline{\mathtt{T}}\rangle\ \mathtt{ok} \qquad \Delta; \Gamma \vdash \overline{\mathtt{M}}\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{T}}\rangle}{\Delta; \Gamma \vdash \mathtt{new}\ \mathtt{I}\langle\overline{\mathtt{T}}\rangle()\{\overline{\mathtt{M}}\} : \mathtt{I}\langle\overline{\mathtt{T}}\rangle} \qquad (\text{GT-ANONYMNEW}_{\text{FGAJ}})$$

$$\frac{\Delta; \Gamma \vdash \mathtt{a} \downarrow \mathtt{T} \qquad \mathtt{e} = \mathtt{a} : \mathtt{T}}{\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}} \qquad (\text{GT-CLOSURE}_{\text{FGATCJ}})$$

$$\frac{\mathtt{T} = \mathtt{I}\langle\overline{\mathtt{T}}\rangle \qquad \Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}}{\Delta; \Gamma \vdash (\mathtt{T})\mathtt{e} : \mathtt{T}} \qquad (\text{GT-CCAST}_{\text{FGATCJ}})$$

1. The judgement for a (generic) type `T` (see **Table 5**) has the form $\Delta \vdash \mathtt{T}\ \mathtt{ok}$ meaning that "`T` is a well-formed type in the type environment $\Delta$".
2. The judgement for sub-typing (see **Table 5**) has the form $\Delta \vdash \mathtt{S} <: \mathtt{T}$ meaning that "`S` is a subtype of `T` in $\Delta$".
3. The judgement for classes (see rule GT-CLASS$_{\text{FGJ}}$ in **Table 4DNb**) has the form `C OK` meaning that "`C` is well typed".
4. Similarly, the judgement for interfaces (see rule GT-INTERF$_{\text{FGAJ}}$ in **Table 4DNb**) has the form `I OK` meaning that "`I` is well typed".
5. The judgement for class methods (see GT-METHOD$_{\text{FGJ}}$ in **Table 4DNb**) has the form `M OK IN C` meaning that "`M` is well typed when its declaration occurs in class `C`". The same judgement is used for method signatures in interfaces (see GT-HEADER$_{\text{FGAJ}}$ in **Table 4DNb**) where `H OK IN I` means that "`H` is a well typed signature in interface `I`".
6. The judgement for methods of interface instances is $\Delta; \Gamma \vdash \mathtt{M}\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{V}}\rangle$ meaning that "`M` is well typed when its declaration occurs in interface instance $\mathtt{I}\langle\overline{\mathtt{V}}\rangle$ which is a well typed anonymous class instance in the context of environments $\Delta$ and $\Gamma$" (see GT-ANONYM$_{\text{FGAJ}}$ in **Table 4DNb**).
7. The judgement for expressions (see the rules of **Table 4DN**) has the form $\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$ meaning that expression "`e` has type `T` in the context of environments $\Delta$ and $\Gamma$".
8. The judgment for SAM typed closures (see the rule of **Table 4DNa**) has the form $\Delta; \Gamma \vdash\ a \downarrow \mathtt{T}$ meaning that closure "$a$ is compatible with SAM type `T` in the context of environments $\Delta$ and $\Gamma$".

**Table 4DNa: Declarative Typing Rule - target compatibility**

**Closure compatibility**

$$\frac{\begin{array}{c} bound_{\Delta}(\mathtt{T}) = \mathtt{I}\langle\overline{\mathtt{V}}\rangle \qquad \Delta \vdash \mathtt{Fun}(\mathtt{I}\langle\overline{\mathtt{V}}\rangle) \qquad \mathtt{a} = \langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle(\overline{\mathtt{T}}\,\overline{\mathtt{x}}) \rightarrow \mathtt{e} \\ \Delta \vdash \mathit{met}(\mathtt{I}\langle\overline{\mathtt{V}}\rangle) = \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{U}\ \mathtt{m}(\overline{\mathtt{S}}\,\overline{\mathtt{w}}) \qquad \overline{\mathtt{N}} = [\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\overline{\mathtt{P}} \\ \Delta \vdash [\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\overline{\mathtt{S}} <: \overline{\mathtt{T}} \qquad \Delta; \Gamma, \overline{\mathtt{x}} : \overline{\mathtt{T}} \vdash \mathtt{e} : \mathtt{Z} \qquad \Delta \vdash \mathtt{Z} <: [\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\mathtt{U} \end{array}}{\Delta; \Gamma \vdash \mathtt{a} \downarrow \mathtt{T}} \qquad (\text{COMP.T}_{\text{FGATCJ}})$$

**Table 4DNb: Typing Rules**

**Classes, Interfaces, Methods**

$$\frac{\begin{array}{c} \Delta = \overline{\mathtt{X}} <: \overline{\mathtt{N}},\ \overline{\mathtt{Y}} <: \overline{\mathtt{P}} \qquad \Delta \vdash \overline{\mathtt{T}}, \mathtt{T}, \overline{\mathtt{P}}\ \mathtt{ok} \\ \Delta; \overline{\mathtt{x}} : \overline{\mathtt{T}}, \mathtt{this} : \mathtt{C}\langle\overline{\mathtt{X}}\rangle \vdash \mathtt{e}_0 : \mathtt{S} \qquad \Delta \vdash \mathtt{S} <: \mathtt{T} \\ \mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\{\ldots\} \qquad \mathtt{override}(\mathtt{m}, \mathtt{N}, \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\overline{\mathtt{T}} \mapsto \mathtt{T}) \end{array}}{\langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}})\{\uparrow \mathtt{e}_0;\}\ \mathtt{OK\ IN}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle} \qquad (\text{GT-METHOD}_{\text{FGJ}})$$

$$\frac{\overline{\mathtt{Y}} <: \overline{\mathtt{P}}, \overline{\mathtt{X}} <: \overline{\mathtt{N}} \vdash \overline{\mathtt{T}}, \mathtt{T}, \overline{\mathtt{P}}\ \mathtt{ok}}{\langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}})\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle} \qquad (\text{GT-HEADER}_{\text{FGAJ}})$$

$$\frac{\begin{array}{c} \Delta' = \Delta, \overline{\mathtt{Y}} <: \overline{\mathtt{P}} \qquad \Delta' \vdash \overline{\mathtt{T}}, \mathtt{T}, \overline{\mathtt{P}}\ \mathtt{ok} \\ \Delta'; \Gamma, \overline{\mathtt{x}} : \overline{\mathtt{T}}, \mathtt{this} : \mathtt{I}\langle\overline{\mathtt{V}}\rangle \vdash \mathtt{e}_0 : \mathtt{S} \qquad \overline{\mathtt{P}}' = [\overline{\mathtt{Y}}'/\overline{\mathtt{Y}}][\overline{\mathtt{V}}/\overline{\mathtt{X}}]\overline{\mathtt{P}} \qquad \Delta' \vdash \mathtt{S} <: \mathtt{T} \\ \mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\{\overline{\mathtt{H}}\} \qquad \Delta \vdash \overline{\mathtt{V}} <: [\overline{\mathtt{V}}/\overline{\mathtt{X}}]\overline{\mathtt{N}} \qquad \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}}) \in \overline{\mathtt{H}} \end{array}}{\Delta; \Gamma \vdash \langle\overline{\mathtt{Y}}' \lhd \overline{\mathtt{P}}'\rangle\mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}})\{\uparrow \mathtt{e}_0;\}\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{V}}\rangle} \qquad (\text{GT-ANONYM}_{\text{FGAJ}})$$

$$\frac{\begin{array}{c} \overline{\mathtt{X}} <: \overline{\mathtt{N}} \vdash \overline{\mathtt{N}}, \mathtt{N}, \overline{\mathtt{T}}\ \mathtt{ok} \qquad \overline{\mathtt{M}}\ \mathtt{OK\ IN}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \\ fields(\mathtt{N}) = \overline{\mathtt{U}}\,\overline{\mathtt{g}} \qquad \mathtt{K} = \mathtt{C}(\overline{\mathtt{U}}\,\overline{\mathtt{g}}, \overline{\mathtt{T}}\,\overline{\mathtt{f}})\{\text{super}(\overline{\mathtt{g}});\ \text{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};\} \end{array}}{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\ \{\overline{\mathtt{T}}\,\overline{\mathtt{f}}; \mathtt{K}\ \overline{\mathtt{M}}\}\ \mathtt{OK}} \qquad (\text{GT-CLASS}_{\text{FGJ}})$$

$$\frac{\overline{\mathtt{X}} <: \overline{\mathtt{N}} \vdash \overline{\mathtt{N}}\ \mathtt{ok} \quad \overline{\mathtt{H}}\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle}{\mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\{\overline{\mathtt{H}}\}\ \mathtt{OK}} \qquad (\text{GT-INTERF}_{\text{FGAJ}})$$

**Table 5: Subtypes**

| | |
|---|---|
| **Subtypes** | |
| $bound_{\Delta}(\mathtt{X}) = \Delta(\mathtt{X})$ | (B-VAR$_{\text{FGJ}}$) |
| $bound_{\Delta}(\mathtt{N}) = \mathtt{N}$ | (B-CLASS$_{\text{FGJ}}$) |
| $bound_{\Delta}(\mathtt{I}\langle\overline{\mathtt{V}}\rangle) = \mathtt{I}\langle\overline{\mathtt{V}}\rangle$ | (B-INTERFACE$_{\text{FGJ}}$) |
| $\Delta \vdash \mathtt{T} <: \mathtt{T}$ | (S-REFL$_{\text{FGJ}}$) |
| $\dfrac{\Delta \vdash \mathtt{S} <: \mathtt{T} \qquad \Delta \vdash \mathtt{T} <: \mathtt{U}}{\Delta \vdash \mathtt{S} <: \mathtt{U}}$ | (S-TRANS$_{\text{FGJ}}$) |
| $\Delta \vdash \mathtt{X} <: \Delta(\mathtt{X})$ | (S-VAR$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\{\ldots\}}{\Delta \vdash \mathtt{C}\langle\overline{\mathtt{T}}\rangle <: [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{N}}$ | (S-CLASS$_{\text{FGJ}}$) |
| **Well-formed types** | |
| $\Delta \vdash \text{Object}\ \mathtt{ok}$ | (WF-OBJECT$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{X} \in dom(\Delta)}{\Delta \vdash \mathtt{X}\ \mathtt{ok}}$ | (WF-VAR$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\{\ldots\} \qquad \Delta \vdash \overline{\mathtt{T}}\ \mathtt{ok} \qquad \Delta \vdash \overline{\mathtt{T}} <: [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{N}}}{\Delta \vdash \mathtt{C}\langle\overline{\mathtt{T}}\rangle\ \mathtt{ok}}$ | (WF-CLASS$_{\text{FGJ}}$) |
| $\dfrac{\mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\{\ldots\} \qquad \Delta \vdash \overline{\mathtt{T}}\ \mathtt{ok} \qquad \Delta \vdash \overline{\mathtt{T}} <: [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{N}}}{\Delta \vdash \mathtt{I}\langle\overline{\mathtt{T}}\rangle\ \mathtt{ok}}$ | (WF-INTERF$_{\text{FGAJ}}$) |

The typing rules are contained in **Table 4DN** and extend those of FGJ [11], and those of FGAJ [3]. The new rules define the type for closures and casting on closures. They are very simple, the first one says that a closure $\mathtt{a} : \mathtt{T}$ has type $\mathtt{T}$ provided that $\Delta; \Gamma \vdash \mathtt{a} \downarrow \mathtt{T}$. The second one says that the type of the closure must be the same as the type to which the closure is cast.

## 4.1. Properties of the Type System

We prove the soundness of the declarative type system of FGATCJ: It guarantees that programs that are well-typed have computations that if terminate then either result into a value or get stuck at a failing type cast. Analogously to [11], we prove the subject reduction theorem and the progress theorem first, the type soundness immediately follows. For space problems, the complete theorem proofs are deferred to the extended version [4].

**Theorem 4.1. (Subject reduction)**
If $\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$ and $\mathtt{e} \rightsquigarrow \mathtt{e}'$ then $\Delta; \Gamma \vdash \mathtt{e}' : \mathtt{T}'$, for some $\mathtt{T}'$ such that $\Delta \vdash \mathtt{T}' <: \mathtt{T}$.

**Proof:**
By induction on the reduction $\mathtt{e} \rightsquigarrow \mathtt{e}'$, with a case analysis on the reduction rule used. The proof of the corresponding theorem for FGJ (pp. 426-428, [11]), has been extended to include rules GR-CLOS-INV-TYPE and GR-CCAST □

Let $\mathtt{e}$ be an expression. Then, $\mathtt{e}$ is a *well-typed* expression if and only if $\Delta, \Gamma, \mathtt{T}$ exist such that: $\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$. Moreover, $\mathtt{e}$ is a *closed, well-typed,* expression when, in addition, $\Delta = \emptyset, \Gamma = \emptyset$. A class $\mathtt{C}$, respectively an interface $\mathtt{I}$, is *well-typed* if and only if $\mathtt{C}\ \mathtt{ok}$, respectively $\mathtt{I}\ \mathtt{ok}$ holds. A program is well-typed if and only if its classes, interfaces, and expressions are all well-typed.

**Theorem 4.2. (Progress)**
Suppose $\mathtt{e}$ is well-typed. If $\mathtt{e}$ includes as a subexpression:
1. $\mathtt{new\ N(\overline{e}).f}$ then $fields(\mathtt{N}) = \overline{\mathtt{T}}\,\overline{\mathtt{f}}$, for some $\overline{\mathtt{T}}$ and $\overline{\mathtt{f}}$, and $\mathtt{f} \in \overline{\mathtt{f}}$.
2. $\mathtt{new\ N(\overline{e}).m\langle\overline{V}\rangle(\overline{d})}$ then $mbody(\mathtt{m}\langle\overline{\mathtt{V}}\rangle, \mathtt{N}) = \overline{\mathtt{x}}.\mathtt{e}_0$, for some $\overline{\mathtt{x}}$ and $\mathtt{e}_0$, and $|\overline{\mathtt{x}}| = |\overline{\mathtt{d}}|$.
3. $(\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle(\overline{\mathtt{T}\ \mathtt{x}}) \rightarrow \mathtt{e}_0 : \mathtt{T}).\mathtt{m}\langle\overline{\mathtt{S}}\rangle(\overline{\mathtt{d}})$ then $|\overline{\mathtt{x}}| = |\overline{\mathtt{d}}| = |\overline{\mathtt{T}}|$ for some $\overline{\mathtt{T}}$, $\overline{\mathtt{x}}$ and $\mathtt{e}_0$.

**Proof:**
The proof is based on the analysis of all well-typed expressions and concludes that either are normal forms or they fall in one of the above three cases □

**Theorem 4.3. (Type Soundness)**
If $\emptyset; \emptyset \vdash \mathtt{e} : \mathtt{T}$ and $\mathtt{e} \leadsto^* \mathtt{e}'$ with $\mathtt{e}'$ a normal form, then $\mathtt{e}'$ either is a value $\mathtt{v} \in \mathcal{V}$ with $\emptyset; \emptyset \vdash \mathtt{v} : \mathtt{S}$ and $\emptyset \vdash \mathtt{S} <: \mathtt{T}$, or an expression containing $(\mathtt{P})(\mathtt{new\ N}(\overline{\mathtt{e}}))$ with $\mathtt{N} \not<: \mathtt{P}$

**Proof:**
By Theorem 4.2, if $\mathtt{e}$ is well-typed then either it is a normal form expression or it is not. In the last case, $\mathtt{e}$ (contains a sub-expression) that can be reduced and results into an expression $\mathtt{e}'$ that, by Theorem 4.1, is a well-typed expression, again. In the first case, we observe that either $\mathtt{e} = \mathtt{e}' \in \mathcal{V}$, or $\mathtt{e}$ is stuck in a type cast, since they are the only well typed, closed, normal form, expressions □

## 5. Type Inference

In this case the program concrete syntax may omit to specify, in closure definitions, the following type annotations:
(a) all or some parameter types;
(b) the closure type;
Hence, the aim of type inference is to find a type assignment for the omitted annotations, that guarantees the correct type checking of the resulting program. In order to do it, the program abstract syntax replaces the omitted annotations with a special class of types. Hence, we need to extend typing in two ways. Firstly, we need a new kind of variable for types. This kind of type variable will be called dotted variable (d-variable, for short), it will be denoted by a variable identifier, $\mathtt{X}$, preceded by $\bullet$, and differs from generic variables because it is used as a placeholder for those type annotations that are omitted in the program.

**Table 4INR - Types, Type Constraints and** FGATCJ$^\bullet$

| Types | Type Constraints |
|---|---|
| $\mathtt{T_R} ::= \mathtt{T} \mid \bullet\mathtt{X}$<br>$\mathtt{T} ::= \mathtt{X} \mid \mathtt{C}\langle\overline{\mathtt{T}}\rangle \mid \mathtt{I}\langle\overline{\mathtt{T}}\rangle$ | $\mathcal{R} ::= \mathtt{T_R} = \mathtt{T_R} \mid \mathtt{T_R} <_\bullet \mathtt{T_R} \mid \mathcal{R} \cup \mathcal{R}$ |
| TC$^\bullet$: *Targeted Closures with omitted type annotations*<br>$\mathtt{e} ::= \mathtt{a} : \mathtt{T_R} \mid (\mathtt{I}\langle\overline{\mathtt{T}}\rangle)\mathtt{e}$<br>$\mathtt{a} ::= \langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle(\overline{\mathtt{T_R}}\ \overline{\mathtt{x}}) \rightarrow \mathtt{e}$ | |
| FGATCJ$^\bullet$=FGJ + IA + TC$^\bullet$ | |

The second extension is concerned with the constraints that are generated by the type inference system, when it collects the requirements that the omitted type annotations should satisfy. This is reported in **Table 4INR**: The syntax $\mathtt{T_R}$ of types which extends $\mathtt{T}$, and the syntax $\mathcal{R}$ of constraints. For notational convenience, $\mathcal{R}_1 \cup ... \cup \mathcal{R}_n$ is represented by (and called) a constraint sequence $\overline{\mathcal{R}}$.

Let $\mathcal{S}^{\mathtt{T}}$, respectively $\mathcal{S}^{\mathtt{T_R}}$, be the set of terms (types) of $\mathtt{T}$, respectively $\mathtt{T_R}$. Let $\mathcal{S}^{\mathtt{X}\bullet}$ be the denumerable set of d-variables, and FGATCJ$^\bullet$ be the set of all the terms of the language FGATCJ extended by allowing d-variables in the type annotations. The type inference system consists of a set of rules defining constrained judgements. The new rules are contained in tables **Table 4IN**, **4INa** and **4INb**. Each rule in **Table 4IN** (and similarly for the other two) has a corresponding rule in **Table 4DN** (**4DNa, 4DNb**) and uses the corresponding constrained judgement. For instance, the constrained judgement for expressions has the form $\Delta;\Gamma\vdash_{\mathtt{inf}} \mathtt{e} : \mathtt{T}\ |\overline{\mathcal{R}}$ meaning that expression $\mathtt{e}$ has type $\mathtt{T}$ in the context $\Delta$ and $\Gamma$, provided that all constraints in $\overline{\mathcal{R}}$ are satisfied (by an assignment of types to d-variables)

## 5.1. Constrained Judgements

**Table 4IN: Inference Typing Rules**

| | |
|---|---|
| $\Delta;\Gamma_1,\mathtt{x}:\mathtt{T},\Gamma_2\vdash_{\mathtt{inf}}\mathtt{x}:\mathtt{T}\vert\ \emptyset$ | (IGT-VAR$_{\mathrm{FGJ}}$) |
| $\dfrac{\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}_0:\mathtt{T}_0\vert\ \overline{\mathcal{R}}\qquad \mathit{fields}(\mathit{bound}_\Delta(\mathtt{T}_0))=\overline{\mathtt{T}}\ \overline{\mathtt{f}}}{\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}_0.\mathtt{f}_i:\mathtt{T}_i\vert\ \overline{\mathcal{R}}}$ | (IGT-FIELD$_{\mathrm{FGJ}}$) |
| $\dfrac{\begin{array}{c}\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}_0:\mathtt{T}_0\vert\ \overline{\mathcal{R}_0}\qquad\Delta\vdash\overline{\mathtt{V}}\ \mathtt{ok}\\ \mathit{mtype}(\mathtt{m},\mathit{bound}_\Delta(\mathtt{T}_0))=\langle\overline{\mathtt{Y}}\lhd\overline{\mathtt{P}}\rangle\overline{\mathtt{U}}\mapsto\mathtt{U}\qquad\Delta\vdash\overline{\mathtt{V}}<:[\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\overline{\mathtt{P}}\\ \Delta;\Gamma\vdash_{\mathtt{inf}}\overline{\mathtt{e}}:\overline{\mathtt{S}}\vert\ \overline{\mathcal{R}_{\overline{\mathtt{e}}}}\qquad\Delta\vdash\overline{\mathtt{S}}<:[\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\overline{\mathtt{U}}\\ \overline{\mathcal{R}}=\{\overline{\mathtt{S}}<_\bullet[\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\overline{\mathtt{U}}\}\cup\overline{\mathcal{R}_0}\cup\overline{\mathcal{R}_{\overline{\mathtt{e}}}}\end{array}}{\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}_0.\mathtt{m}\langle\overline{\mathtt{V}}\rangle(\overline{\mathtt{e}}):[\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\mathtt{U}\vert\ \overline{\mathcal{R}}}$ | (IGT-INV$_{\mathrm{FGJ}}$) |
| $\dfrac{\begin{array}{c}\Delta\vdash\mathtt{N}\ \mathtt{ok}\qquad\mathit{fields}(\mathtt{N})=\overline{\mathtt{T}}\ \overline{\mathtt{f}}\\ \Delta;\Gamma\vdash_{\mathtt{inf}}\overline{\mathtt{e}}:\overline{\mathtt{S}}\vert\ \overline{\mathcal{R}_{\overline{\mathtt{e}}}}\qquad\Delta\vdash\overline{\mathtt{S}}<:\overline{\mathtt{T}}\\ \overline{\mathcal{R}}=\{\overline{\mathtt{S}}<_\bullet\overline{\mathtt{T}}\}\cup\overline{\mathcal{R}_{\overline{\mathtt{e}}}}\end{array}}{\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{new}\ \mathtt{N}(\overline{\mathtt{e}}):\mathtt{N}\vert\ \overline{\mathcal{R}}}$ | (IGT-NEW$_{\mathrm{FGJ}}$) |
| $\dfrac{\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}:\mathtt{T}\vert\ \overline{\mathcal{R}_{\mathtt{e}}}\qquad\Delta\vdash \mathit{bound}_\Delta(\mathtt{T})<:\mathtt{N}}{\Delta;\Gamma\vdash_{\mathtt{inf}}(\mathtt{N})\mathtt{e}:\mathtt{N}\vert\ \overline{\mathcal{R}_{\mathtt{e}}}}$ | (IGT-UCAST$_{\mathrm{FGJ}}$) |
| $\dfrac{\begin{array}{c}\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}:\mathtt{T}\vert\ \overline{\mathcal{R}_{\mathtt{e}}}\qquad\Delta\vdash\mathtt{N}\ \mathtt{ok}\qquad\Delta\vdash\mathtt{N}<:\mathit{bound}_\Delta(\mathtt{T})\\ \mathtt{N}=\mathtt{C}\langle\overline{\mathtt{T}}\rangle\qquad\mathit{bound}_\Delta(\mathtt{T})=\mathtt{D}\langle\overline{\mathtt{U}}\rangle\qquad\mathit{dcast}(\mathtt{C},\mathtt{D})\end{array}}{\Delta;\Gamma\vdash_{\mathtt{inf}}(\mathtt{N})\mathtt{e}:\mathtt{N}\vert\ \overline{\mathcal{R}_{\mathtt{e}}}}$ | (IGT-DCAST$_{\mathrm{FGJ}}$) |
| $\dfrac{\begin{array}{c}\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}:\mathtt{T}\vert\ \overline{\mathcal{R}_{\mathtt{e}}}\qquad\Delta\vdash\mathtt{N}\ \mathtt{ok}\\ \mathtt{N}=\mathtt{C}\langle\overline{\mathtt{T}}\rangle\qquad\mathit{bound}_\Delta(\mathtt{T})=\mathtt{D}\langle\overline{\mathtt{U}}\rangle\qquad\mathtt{C}\not\trianglelefteq\mathtt{D}\qquad\mathtt{D}\not\trianglelefteq\mathtt{C}\end{array}}{\Delta;\Gamma\vdash_{\mathtt{inf}}(\mathtt{N})\mathtt{e}:\mathtt{N}\vert\ \overline{\mathcal{R}_{\mathtt{e}}}}$ | (IGT-SCAST$_{\mathrm{FGJ}}$) |
| $\dfrac{\Delta\vdash\mathtt{I}\langle\overline{\mathtt{T}}\rangle\ \mathtt{ok}\qquad\Delta;\Gamma\vdash\overline{\mathtt{M}}\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{T}}\rangle}{\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{new}\ \mathtt{I}\langle\overline{\mathtt{T}}\rangle()\{\overline{\mathtt{M}}\}:\mathtt{I}\langle\overline{\mathtt{T}}\rangle\vert\ \emptyset}$ | (IGT-ANONYMNEW$_{\mathrm{FGAJ}}$) |
| $\dfrac{\begin{array}{c}\mathtt{e}=\mathtt{a}:\mathtt{T}_a\qquad\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{a}\downarrow\mathtt{T}\vert\ \overline{\mathcal{R}_{\mathtt{a}}}\\ \overline{\mathcal{R}}=\{\mathtt{T}_a=\mathtt{T}\}\cup\overline{\mathcal{R}_{\mathtt{a}}}\end{array}}{\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}:\mathtt{T}\ \vert\ \overline{\mathcal{R}}}$ | (IGT-CLOSURE$_{\mathrm{FGATCJ}}$) |
| $\dfrac{\mathtt{T}=\mathtt{I}\langle\overline{\mathtt{T}}\rangle\qquad\Delta;\Gamma\vdash_{\mathtt{inf}}\mathtt{e}:\mathtt{T}\vert\ \overline{\mathcal{R}_{\mathtt{e}}}}{\Delta;\Gamma\vdash_{\mathtt{inf}}(\mathtt{T})\mathtt{e}:\mathtt{T}\ \vert\ \overline{\mathcal{R}_{\mathtt{e}}}}$ | (IGT-CCAST$_{\mathrm{FGATCJ}}$) |

Constrained judgments are essentially the judgements of the declarative typing system, extended with the constraints of a suitable constraint system $(\Sigma,\mathcal{A},\mathcal{X},\mathcal{L})^2$ [12], or more simply, of a unification problem $(\Sigma,\mathcal{X},E)$ [8, 5], where the logic $\mathcal{L}$ is replaced by an algebra $E$, having a possibly complete, rewriting system which conveniently, provides the core of the constraint system solver. However, the algebra $E$ results uselessly tricky in order to deal with constraints $\mathtt{T}_1<_\bullet\mathtt{T}_2$. Hence, we prefer treating the constraint

[2] $\Sigma=\{\mathtt{X}^0,\mathtt{C}^n,\mathtt{I}^n\vert\ n\in\aleph\}^3$ is the signature of type constructors; $\mathcal{A}=\mathcal{S}^{\mathtt{T}}$ is the constraint computation domain; $\mathcal{X}=\mathcal{S}^{\mathtt{X}\bullet}$ is the set of variables; $\mathcal{L}$ is the set of the admitted formulas.

system as an ordinary, first-order unification [9] on $\mathcal{S}^{\mathtt{T}_\mathtt{R}}$, where: Constraint $\mathtt{T}_1 = \mathtt{T}_2$ is asserting that $\mathtt{T}_1$ and $\mathtt{T}_2$ must be unified, whilst $\mathtt{T}_1 <_\bullet \mathtt{T}_2$ is asserting that $\mathtt{T}_1$ and $\mathtt{T}_2$ must be unified, only if one of the two is bound to a d-variable. In fact, constraints of the form $\mathtt{T}_1 <_\bullet \mathtt{T}_2$ come from the presence of subtyping in the declarative type system. This treatment of such constraints reflects the simplificative choice, of keeping the subtyping rules outside the type inference.

**Table 4INa: Inference Typing Rule - target compatibility**

**Closure compatibility**

$$\dfrac{\begin{array}{c} bound_\Delta(\mathtt{T}) = \mathtt{I}\langle\overline{\mathtt{V}}\rangle \qquad \mathtt{a} = \langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle(\overline{\mathtt{T}}\,\overline{\mathtt{x}}) \to \mathtt{e} \\ \Delta \vdash \mathit{met}(\mathtt{I}\langle\overline{\mathtt{V}}\rangle) = \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{U}\,\mathtt{m}(\overline{\mathtt{S}}\,\overline{\mathtt{w}}) \qquad \overline{\mathtt{N}} = [\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\overline{\mathtt{P}} \qquad \Delta \vdash \mathtt{Fun}(\mathtt{I}\langle\overline{\mathtt{V}}\rangle) \\ \Delta \vdash [\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\overline{\mathtt{S}} <: \overline{\mathtt{T}} \qquad \Delta; \Gamma, \overline{\mathtt{x}} : \overline{\mathtt{T}} \vdash_{\mathtt{inf}} \mathtt{e} : \mathtt{T}_e \mid \overline{\mathcal{R}}_\mathtt{e} \qquad \Delta \vdash \mathtt{T}_e <: [\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\mathtt{U} \\ \overline{\mathcal{R}} = \{[\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\overline{\mathtt{S}} <_\bullet \overline{\mathtt{T}};\ \mathtt{T}_e <_\bullet [\overline{\mathtt{X}}/\overline{\mathtt{Y}}]\mathtt{U}\} \cup \overline{\mathcal{R}}_e \end{array}}{\Delta; \Gamma \vdash_{\mathtt{inf}} \mathtt{a} \downarrow \mathtt{T} \mid \overline{\mathcal{R}}} \qquad (\textsc{IComp.T}_{\text{FGATCJ}})$$

**Table 4INb: Typing Rules**

**Classes, Interfaces, Methods**

$$\dfrac{\begin{array}{c} \Delta = \overline{\mathtt{X}} <: \overline{\mathtt{N}},\ \overline{\mathtt{Y}} <: \overline{\mathtt{P}} \qquad \Delta \vdash \overline{\mathtt{T}}, \mathtt{T}, \overline{\mathtt{P}}\ \mathtt{ok} \\ \Delta; \overline{\mathtt{x}} : \overline{\mathtt{T}}, \mathtt{this} : \mathtt{C}\langle\overline{\mathtt{X}}\rangle \vdash_{\mathtt{inf}} \mathtt{e}_0 : \mathtt{S} \mid \overline{\mathcal{R}} \qquad \mathtt{solved}(\overline{\mathcal{R}}) \qquad \Delta \vdash \mathtt{S} <: \mathtt{T} \\ \mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\{\ldots\} \qquad \mathtt{override}(\mathtt{m}, \mathtt{N}, \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\overline{\mathtt{T}} \mapsto \mathtt{T}) \end{array}}{\langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}})\{\uparrow \mathtt{e}_0;\}\ \mathtt{OK\ IN}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle} \qquad (\textsc{GT-Method}_{\text{FGJ}})$$

$$\dfrac{\overline{\mathtt{Y}} <: \overline{\mathtt{P}}, \overline{\mathtt{X}} <: \overline{\mathtt{N}} \vdash \overline{\mathtt{T}}, \mathtt{T}, \overline{\mathtt{P}}\ \mathtt{ok}}{\langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}})\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle} \qquad (\textsc{GT-Header}_{\text{FGAJ}})$$

$$\dfrac{\begin{array}{c} \Delta' = \Delta, \overline{\mathtt{Y}} <: \overline{\mathtt{P}} \qquad \Delta' \vdash \overline{\mathtt{T}}, \mathtt{T}, \overline{\mathtt{P}}\ \mathtt{ok} \\ \Delta'; \Gamma, \overline{\mathtt{x}} : \overline{\mathtt{T}}, \mathtt{this} : \mathtt{I}\langle\overline{\mathtt{V}}\rangle \vdash_{\mathtt{inf}} \mathtt{e}_0 : \mathtt{S} \mid \overline{\mathcal{R}} \qquad \mathtt{solved}(\overline{\mathcal{R}}) \qquad \mathtt{P}' = [\mathtt{Y}'/\mathtt{Y}][\overline{\mathtt{V}}/\mathtt{X}]\mathtt{P} \\ \Delta' \vdash \mathtt{S} <: \mathtt{T} \qquad \mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\{\overline{\mathtt{H}}\} \qquad \langle\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\rangle\mathtt{T}\,\mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}}) \in \overline{\mathtt{H}} \end{array}}{\Delta; \Gamma \vdash \langle\overline{\mathtt{Y}}' \lhd \overline{\mathtt{P}}'\rangle\mathtt{T}\,\mathtt{m}(\overline{\mathtt{T}}\,\overline{\mathtt{x}})\{\uparrow \mathtt{e}_0;\}\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{V}}\rangle} \qquad (\textsc{GT-Anonym}_{\text{FGAJ}})$$

$$\dfrac{\begin{array}{c} \overline{\mathtt{X}} <: \overline{\mathtt{N}} \vdash \overline{\mathtt{N}}, \mathtt{N}, \overline{\mathtt{T}}\ \mathtt{ok} \qquad \overline{\mathtt{M}}\ \mathtt{OK\ IN}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \\ fields(\mathtt{N}) = \overline{\mathtt{U}}\,\overline{\mathtt{g}} \qquad \mathtt{K} = \mathtt{C}(\overline{\mathtt{U}}\,\overline{\mathtt{g}}, \overline{\mathtt{T}}\,\overline{\mathtt{f}})\{\text{super}(\overline{\mathtt{g}});\ \text{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};\} \end{array}}{\mathtt{class}\ \mathtt{C}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle \lhd \mathtt{N}\ \{\overline{\mathtt{T}}\,\overline{\mathtt{f}}; \mathtt{K}\,\overline{\mathtt{M}}\}\ \mathtt{OK}} \qquad (\textsc{GT-Class}_{\text{FGJ}})$$

$$\dfrac{\overline{\mathtt{X}} <: \overline{\mathtt{N}} \vdash \overline{\mathtt{N}}\ \mathtt{ok} \qquad \overline{\mathtt{H}}\ \mathtt{OK\ IN}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle}{\mathtt{interface}\ \mathtt{I}\langle\overline{\mathtt{X}} \lhd \overline{\mathtt{N}}\rangle\{\overline{\mathtt{H}}\}\ \mathtt{OK}} \qquad (\textsc{GT-Interf}_{\text{FGAJ}})$$

**Auxiliary predicates**

$$\dfrac{\mathtt{Sols}(\overline{\mathcal{R}}) \neq \{\}}{\mathtt{solved}(\overline{\mathcal{R}})} \qquad (\overline{\mathcal{R}}\textsc{-Solver})$$

## 5.2. Properties of the Type Inference System

For all terms $\mathtt{u} \in$ FGATCJ$^\bullet$, $\mathtt{dVar}(\mathtt{u})$ is the set of d-variables that are in $\mathtt{u}$. A (idempotent[4]. Hence, we consider only idempotent substitutions) substitution $\rho$, on $\mathcal{S}^{\mathtt{T}_\mathtt{R}}$ is a (idempotent) function from d-variables into $\mathcal{S}^{\mathtt{T}_\mathtt{R}}$, that maps identically except for a finite set of d-variables, called the domain of $\rho$ and denoted by $\mathtt{dom}(\rho)$. The set $\mathtt{Im}(\rho)$, is the image set of $\rho$ and, the set $\mathtt{dVar}(\rho)$, is the set of all d-variables occurring in the image set of $\rho$. Each substitution $\rho$ on $\mathcal{S}^{\mathtt{T}_\mathtt{R}} \subset$ FGATCJ$^\bullet$ is uniquely, extended to a endomorphism $\rho^E$ on FGATCJ$^\bullet$: For each $\mathtt{u} \in$ FGATCJ$^\bullet$, $\rho^E(\mathtt{u})$ is the term of FGATCJ$^\bullet$ resulting by replacing, in $\mathtt{u}$, $\bullet\mathtt{X}$ with $\rho(\bullet\mathtt{X})$, for all d-variables. Let $\mathtt{Subs}(\mathcal{S}^{\mathtt{T}_\mathtt{R}})$ be the set of all the substitutions on $\mathcal{S}^{\mathtt{T}_\mathtt{R}}$. $\mathtt{Subs}(\mathcal{S}^{\mathtt{T}_\mathtt{R}})$ contains the identity substitution $\varepsilon$, it is closed under function composition $\circ$, i.e.

[4]We consider here, only idempotent substitutions that furnish a compact and simple, algebraic framework [9], for term unification and constraint solving on terms

$\rho \circ \sigma \in \mathtt{Subs}(\mathcal{S}^{\mathtt{T}_\mathtt{R}})$ for all $\rho, \sigma \in \mathtt{Subs}(\mathcal{S}^{\mathtt{T}_\mathtt{R}})$, it has a partial ordering $\preceq$ having $\varepsilon$ at the bottom. Identity is an idempotent substitution and has $\mathtt{dom}(\varepsilon) = \mathtt{Im}(\varepsilon) = \mathtt{dVar}(\varepsilon)$ since all are the emptyset.

**Definition 5.1. ($\mathtt{dom}$, $\mathtt{Im}$, $\mathtt{dVar}$, $\varepsilon$, $\preceq$)**
$\mathtt{dom}(\rho) = \{\bullet\mathtt{X} | \rho(\bullet\mathtt{X}) \neq \bullet\mathtt{X}\}$;
$\mathtt{Im}(\rho) = \{\rho(\bullet\mathtt{X}) | \bullet\mathtt{X} \in \mathtt{dom}(\rho)\}$;
$\mathtt{dVar}(\rho) = \bigcup_{\mathtt{u} \in \mathtt{Im}(\rho)} \mathtt{dVar}(\mathtt{u})$;
$\varepsilon(\bullet\mathtt{X}) = \bullet\mathtt{X}, \forall \bullet \mathtt{X} \in \mathcal{S}^{\bullet\mathtt{X}}$;
$\rho \preceq \sigma$ if and only if $\sigma = \delta \circ \rho$ for some $\delta$.

To each $\overline{\mathcal{R}}$ corresponds a finite set of finite sets of terms $\mathcal{M}(\overline{\mathcal{R}})$ such that a solution of $\overline{\mathcal{R}}$ is any unifier [5] $\rho$ of $\mathcal{M}(\overline{\mathcal{R}})$. Hence, the solution set $\mathtt{Sols}(\overline{\mathcal{R}})$ of a constraint sequence $\overline{\mathcal{R}}$ is the set of all the unifiers of $\mathcal{M}(\overline{\mathcal{R}})$. Let $\mathtt{mgu}$ be the most general unifier [6] of a set of sets of terms on $\mathcal{S}^{\mathtt{T}_\mathtt{R}}$, the most general solver $\rho[\overline{\mathcal{R}}]$, if any, of a sequence $\overline{\mathcal{R}}$ of constraints, is such that $\rho[\overline{\mathcal{R}}] = \mathtt{mgu}(\mathcal{M}(\overline{\mathcal{R}}))$.

**Definition 5.2. ($\mathcal{M}$, $\rho[\overline{\mathcal{R}}]$, $\mathtt{Sols}$)**
$\mathcal{M}(\overline{\mathcal{R}}) = \{\{\mathtt{T}_1, \mathtt{T}_2\} | \mathtt{T}_1 = \mathtt{T}_2 \in \overline{\mathcal{R}}\} \cup \{\{\mathtt{T}_1, \mathtt{T}_2\} | \mathtt{T}_1 <_\bullet \mathtt{T}_2 \in \overline{\mathcal{R}} \text{ and } \{\mathtt{T}_1, \mathtt{T}_2\} \cap \mathcal{S}^{\bullet\mathtt{X}} \neq \{\}\}$;
- $\rho[\overline{\mathcal{R}}] = \mathtt{mgu}(\mathcal{M}(\overline{\mathcal{R}}))$;
- $\mathtt{Sols}(\overline{\mathcal{R}}) = \{\sigma \circ \rho[\overline{\mathcal{R}}] \mid \sigma \in \mathtt{Subs}(\mathcal{S}^{\mathtt{T}_\mathtt{R}})\}$.

**Theorem 5.3. (Soundness)**
For all $\Delta, \Gamma, \mathtt{e}, \mathtt{T}, \overline{\mathcal{R}}$, If:
- $\Delta; \Gamma \vdash_{\mathtt{inf}} \mathtt{e} : \mathtt{T} | \overline{\mathcal{R}}$ and,
- $\rho \in \mathtt{Sols}(\overline{\mathcal{R}})$ and,
- $\mathtt{dVar}(\mathtt{T}) \cup \mathtt{dVar}(\mathtt{e}) \subseteq \mathtt{dom}(\rho)$ and,
- $\mathtt{dVar}(\rho) = \{\}$

Then, $(*)\ \Delta; \rho^E(\Gamma) \vdash \rho^E(\mathtt{e}) : \rho^E(\mathtt{T})$.

**Proof:**
By induction on the size of expression $\mathtt{e}$, measured as the maximum of the sub-term nesting level and case analysis on the last rule used in the inference $\Delta; \Gamma \vdash_{\mathtt{inf}} \mathtt{e} : \mathtt{T} | \overline{\mathcal{R}}$. □

Let $\mathtt{e}^\bullet \in$ FGATCJ$^\bullet$ be an expression. A type assignments for $\mathtt{e}^\bullet$ is any substitution $\rho \in \mathtt{Subs}(\mathcal{S}^{\mathtt{T}_\mathtt{R}})$ such that: $\rho(\mathtt{e}^\bullet) \in$ FGATCJ $\subset$ FGATCJ$^\bullet$ and $\rho(\mathtt{e}^\bullet)$ is correctly typed, i.e.: $\Delta; \Gamma \vdash \rho(\mathtt{e}^\bullet) : \mathtt{T}$ holds for a type $\mathtt{T}$ and well defined $\Delta$ and $\Gamma$.

**Theorem 5.4. (Completeness)**
For all $\Delta, \Gamma, \mathtt{e}, \mathtt{T}$, If:
- $\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$ and,
- $\mathtt{e} = \rho(\mathtt{e}^\bullet)$, for some $\rho \in \mathtt{Subs}(\mathcal{S}^{\mathtt{T}_\mathtt{R}})$, $\mathtt{e}^\bullet \in$ FGATCJ$^\bullet$.

Then, $(**)\ \Delta; \Gamma^\bullet \vdash_{\mathtt{inf}} \mathtt{e}^\bullet : \mathtt{T}^\bullet \mid \overline{\mathcal{R}}$ and $\rho(\Gamma^\bullet) = \Gamma$ and $\rho(\mathtt{T}^\bullet) = \mathtt{T}$.

[5]A unifier [9] is any substitution that makes identical the terms in each term set of $\mathcal{M}(\overline{\mathcal{R}})$
[6]The most general unifier is the lower bound of unifier set, [9] definition 4.7

**Proof:**
By induction on size of expressions e, measured as the maximum of the subterm nesting level and case analysis on the last rule used in the inference $\Delta; \Gamma \vdash \mathtt{e} : \mathtt{T}$. □

## 5.3. Example

Consider a program with three interfaces that define three SAM types. In particular, $\mathtt{I}_0$ and $\mathtt{I}_2$ are such that their closures are compatible both with $\mathtt{I}_0$ and $\mathtt{I}_2$ and consequently they can have both type $\mathtt{I}_0$ and $\mathtt{I}_2$. This is due to the nominal nature of SAM types.

$\mathtt{interface\ I_0\{Integer\ invoke()\};\ interface\ I_1\{I_0\ invoke(Integer\ x)\};}$
$\mathtt{interface\ I_2\{Integer\ invoke()\}}$

We now show how the system infers a unique type for the closure $\mathtt{a} = () \to 3$ when it occurs in a program, for instance the following: $((\mathtt{I}_1)((\mathtt{x}) \to () \to 3)).\mathtt{invoke}(2)$. We start adding dotted variables when needed, hence the expression is rewritten as follows:$((\mathtt{I}_1)((\bullet\mathtt{X}_1\ \mathtt{x}) \to (() \to 3 : \bullet\mathtt{X}_2) : \bullet\mathtt{X}_3)).\mathtt{invoke}(2)$.

The computation is below.

1 $\emptyset;\emptyset \vdash_{\mathtt{inf}} ((\mathtt{I}_1)((\bullet\mathtt{X}_1\ \mathtt{x}) \to (() \to 3) : \bullet\mathtt{X}_2) : \bullet\mathtt{X}_3)).\mathtt{invoke}(2) : \mathtt{T}\ |\overline{\mathcal{R}}$ by IGT-INV

1.1 $\emptyset;\emptyset \vdash_{\mathtt{inf}} (\mathtt{I}_1)((\bullet\mathtt{X}_1\ \mathtt{x}) \to (() \to 3) : \bullet\mathtt{X}_2) : \bullet\mathtt{X}_3) : \mathtt{T}_0|\ \overline{\mathcal{R}}_0$

1.2 $mtype(\mathtt{invoke}, \mathtt{I}_1) = \mathtt{Integer} \to \mathtt{I}_0$ with $\mathtt{T} = \mathtt{I}_0$

1.3 $\emptyset;\emptyset \vdash_{\mathtt{inf}} 2 : \mathtt{Integer}\ |\overline{\mathcal{R}}_e$ with $\overline{\mathcal{R}}_e = \{\}$ since the additional axioms on primitive data

1.4 $\overline{\mathcal{R}} = \{\mathtt{Integer} <_\bullet \mathtt{Integer}\} \cup \overline{\mathcal{R}}_0 \cup \overline{\mathcal{R}}_e$

From 1.1, by IGT-CCAST with $\mathtt{T}_0 = \mathtt{I}_1$

1.1.1 $\emptyset;\emptyset \vdash_{\mathtt{inf}} ((\bullet\mathtt{X}_1\ \mathtt{x}) \to (() \to 3) : \bullet\mathtt{X}_2) : \bullet\mathtt{X}_3) : \mathtt{I}_1|\overline{\mathcal{R}}_{e_1}$
with $\overline{\mathcal{R}}_0 = \overline{\mathcal{R}}_{e_1}$ by IGT-CLOSURE

1.1.1.1 $\emptyset;\emptyset \vdash_{\mathtt{inf}} ((\bullet\mathtt{X}_1\ \mathtt{x}) \to (() \to 3) : \bullet\mathtt{X}_2) \downarrow \mathtt{I}_1\ |\overline{\mathcal{R}}_a$
with $\overline{\mathcal{R}}_{e_1} = \{\bullet\mathtt{X}_3 = \mathtt{I}_1\} \cup \overline{\mathcal{R}}_a$ by ICOMP.T

1.1.1.1.1 $\emptyset \vdash \mathtt{I}_1 <: \mathtt{I}_1$

1.1.1.1.2 $\emptyset \vdash \mathtt{Fun}(\mathtt{I}_1)$

1.1.1.1.3 $\emptyset \vdash met(\mathtt{I}_1) = \mathtt{I}_0\,\mathtt{invoke(Integer\ x)}$

1.1.1.1.4 $\emptyset \vdash \mathtt{T}_{e_1} <: \mathtt{I}_0$

1.1.1.1.5 $\emptyset; \mathtt{x} : \bullet\mathtt{X}_1 \vdash_{\mathtt{inf}} () \to 3 : \bullet\mathtt{X}_2 : \mathtt{T}_{e_1}|\overline{\mathcal{R}}_{e_2}$
with $\overline{\mathcal{R}}_a = \{\mathtt{Integer} <: \bullet\,\mathtt{X}_1, \mathtt{T}_{e_1} <_\bullet \mathtt{I}_0\} \cup \overline{\mathcal{R}}_{e_2}$ by IGT-CLOSURE

1.1.1.1.5.1 $\emptyset; \mathtt{x} : \bullet\mathtt{X}_1 \vdash_{\mathtt{inf}} () \to 3 \downarrow \mathtt{T}_{e_1}|\overline{\mathcal{R}}_{e_3}$ with $\overline{\mathcal{R}}_{e_2} = \{\bullet\mathtt{X}_2 = \mathtt{T}_{e_1}\} \cup \overline{\mathcal{R}}_{e_3}$
By contstraint solving, meta-variable $\mathtt{T}_{e_1}$ is replaced by $\bullet\mathtt{X}_2$ then, $\bullet\mathtt{X}_2$ is replaced by $\mathtt{I}_0$: Hence 1.1.1.1.5.1 becomes
$\emptyset; \mathtt{x} : \bullet\mathtt{X}_1 \vdash_{\mathtt{inf}} () \to 3 \downarrow \mathtt{I}_0|\overline{\mathcal{R}}_{e_3}$ by ICOMP.T

1.1.1.1.5.1.1 $bound(\mathtt{I}_0) = \mathtt{I}_0$

1.1.1.1.5.1.2 $\emptyset \vdash met(\mathtt{I}_0) = \mathtt{Integer\ invoke()}$

1.1.1.1.5.1.3 $\emptyset \vdash \mathtt{Fun}(\mathtt{I}_0)$

1.1.1.1.5.1.4 $\emptyset \vdash \mathtt{T}_{e_2} <: \mathtt{Integer}$

1.1.1.1.5.1.5 $\emptyset; \mathtt{x} : \bullet\mathtt{X}_1 \vdash_{\mathtt{inf}} 3 : \mathtt{Integer} | \overline{\mathcal{R}}_{e_4}$ with $\overline{\mathcal{R}}_{e_4} = \{\}$ since the additional axiom on primitive data

1.1.1.1.5.1.6 $\overline{\mathcal{R}}_{e_3} = \{\mathtt{Integer} <_\bullet \mathtt{Integer}\} \cup \overline{\mathcal{R}}_{e_4}$

$\overline{\mathcal{R}}_{e_2} = \overline{\mathcal{R}}_{e_3} = \{\mathtt{Integer} <_\bullet \mathtt{Integer}\}$

$\overline{\mathcal{R}}_a = \{\mathtt{Integer} <: \bullet\, \mathtt{X}_1, \bullet\mathtt{X}_2 <_\bullet \mathtt{I}_0, \mathtt{Integer} <_\bullet \mathtt{Integer}\}$

$\overline{\mathcal{R}}_{e_1} = \{\bullet\mathtt{X}_3 = \mathtt{I}_1, \mathtt{Integer} <: \bullet\, \mathtt{X}_1, \bullet\mathtt{X}_2 <_\bullet \mathtt{I}_0, \mathtt{Integer} <_\bullet \mathtt{Integer}\}$

$\overline{\mathcal{R}}_0 = \overline{\mathcal{R}}_{e_1} = \{\bullet\mathtt{X}_3 = \mathtt{I}_1, \mathtt{Integer} <: \bullet\, \mathtt{X}_1, \bullet\mathtt{X}_2 <_\bullet \mathtt{I}_0, \mathtt{Integer} <_\bullet \mathtt{Integer}\}$

$\overline{\mathcal{R}} = \{\bullet\mathtt{X}_3 = \mathtt{I}_1, \mathtt{Integer} <: \bullet\, \mathtt{X}_1, \bullet\mathtt{X}_2 <_\bullet \mathtt{I}_0, \mathtt{Integer} <_\bullet \mathtt{Integer}\}$

Hence, the type of the initial expression results by 1.2: $\mathtt{T} = \mathtt{I}_0$, and $\overline{\mathcal{R}}$ is satisfied by the assignment of types to d-variables, $\overline{\mathcal{R}} = \{\bullet\mathtt{X}_3 = \mathtt{I}_1, \bullet\mathtt{X}_1 = \mathtt{Integer}, \bullet\mathtt{X}_2 = \mathtt{I}_0\}$ that is obtained by constraint solving.

## 6. Conclusions

We have provided a type inference system for Java SAM typed closures [6]. It deals with nominal types and furnishes the most general assignment, if one exists, of types to the omitted type annotations of the program. The system has been proved sound and complete. Due to space limitation, the system ignored the mechanism of the interface hierarchy and that of method overloading. However, the approach we have followed is structured enough to allow an easy inclusion of the (computation and typing) rules for the additional mechanisms. Eventually, a type inference algorithm for FGATCJ$^\bullet$ is a program transformation that accepts programs of FGATCJ$^\bullet$ and returns the same program with all expressions annotated with their types, if any, or otherwise, it signals type error. The definition of such an algorithm was out of the scope of the paper, however it has the constrained rules system, introduced in **Tables 4IN**, as its central core, but it needs in addition, to implement a strategy to apply in a deterministic, possibly efficient, way such rules.

## References

[1] D. Lea B. Lee and J. Bloch. Concise Instance Creation Expressions: Closure without Complexity, 2006. crazybob.org/2006/10/java-closure-spectrum.html.

[2] M. Bellia and M.E. Occhiuto. *Java in Academia and Research*, chapter Java$\Omega$: Higher Order Programming in Java, pages 166–185. iConcept Press Ltd., 2011.

[3] M. Bellia and M.E. Occhiuto. The equivalence of Reduction and Translation Semantics of Java Simple Closures. *Fundamenta Informaticae*, 119:1–16, 2012.

[4] M. Bellia and M.E. Occhiuto. Java SAM Typed Closures: A Sound and Complete Type Inference System for Nominal Types(Extended Version). Technical Report TR-13-07, University of Pisa, Dipartimento Informatica, 2013. http://compass2.di.unipi.it/TR/.

[5] F. Baader and K. Schultz. *Cmbining Constraint Solving*, volume 2001 of *LNCS*. Springer-Verlag, 2001.

[6] A. Buckley and D. Smith. *JSR-000335 Lambda Expressions for the Java Programming Language - Early Draft Review: Lambda Specification, Version 0.4.2.* Oracle Corporation, December 2011. http://download.oracle.com/otndocs/jcp/lambda-0_4_2-edr-spec/index.html.

[7] A. Buckley and D. Smith. *State of the Lambda*. Oracle Corporation, December 2011. http://cr.openjdk.java.net/b̃riangoetz/lambda/lambda-state-4.html.

[8] H. Comon and C. Kirchner. *Constraint Solving on Terms*, volume 2001 of *LNCS*. Springer-Verlag, 2001.

[9] Elmar Eder. Properties of Substitutions and Unifications. *J. Symb. Comput.*, 1(1):31–46, March 1985.

[10] Matthias Felleisen and Daniel P. Friedman. A Reduction Semantics for Imperative Higher-Order Languages. In *PARLE (2)*, pages 206–223, 1987.

[11] A. Igarashi, B. Pierce, and P. Wadler. Featherweight Java: A Minimal Core Calculus for Java and GJ. *ACM TOPLAS*, 23:396–450, 2001.

[12] C. Kirchner. *Constraint Solving on Terms: Syntactic Methods. Preliminary Lecture Notes*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.30.6600, 1999.

[13] OpenJDK. Project lambda, 2012. http://openjdk.java.net/projects/lambda/.

[14] B.J. Pierce. *Types and Programming Languages*. MIT Press, 2002.