Recursive Types Are not Conservative over $F_{\leq}$ (*extended abstract*)

Giorgio Ghelli[1]

Abstract. $F_{\leq}$ is a type system used to study the integration of inclusion and parametric polymorphism. $F_{\leq}$ does not include a notion of recursive types, but extensions of $F_{\leq}$ with recursive types are widely used as a basis for foundational studies about the type systems of functional and object-oriented languages. In this paper we show that adding recursive types results in a non conservative extension of the system. This means that the algorithm for $F_{\leq}$ subtyping (the kernel of the algorithm for $F_{\leq}$ typing) is no longer complete for the extended system, even when it is applied only to judgements where no recursive type appears, and that most of the proofs of known properties of $F_{\leq}$ do not hold for the extended system; this is the case, for example, for Pierce's proof of undecidability of $F_{\leq}$. However, we prove that this non conservativity is limited to a very special class of subtyping judgements, the "diverging judgements" introduced in [Ghe]. This last result implies that the extension of $F_{\leq}$ with recursive types could be still useful for practical purposes.

1 Introduction

$F_{\leq}$ is a minimal language integrating subtyping and bounded parametric polymorphism. It is a simplification of the language Fun introduced in [CaWe]; the essential difference is that Fun has recursive types and values, which are missing in $F_{\leq}$. $F_{\leq}$ was introduced in [CuGhe] (see also [BrLo], [CaMaMiSce] and [CaLo]). Extensions of $F_{\leq}$ are being exploited as a foundational tool in the research on the extension of the type system of functional languages with modules or with object-oriented features. We refer to [CaWe], [GheTh], [CuGhe] and [CaMaMiSce] for more motivations and details about $F_{\leq}$.

In all the models of object-oriented type systems, as in many other contexts, recursive types play a central role. It has been argued (see, e.g., [AmCa]) that recursion can be added to $F_{\leq}$ quite painlessly, as a feature which is in some way "orthogonal" to the rest of the system. In this paper we show that this is not the case: the mere "existence" of recursive types changes the subtyping relations between non recursive types.

This result shows that "transitivity elimination" for $F_{\leq}$, in the form proved in [CuGhe], is not valid in $\mu F_{\leq}$. Transitivity elimination means, essentially, that every provable subtyping can be proved without using the transitivity rule below:

$$\frac{\Gamma \vdash T \leq U \qquad \Gamma \vdash U \leq V}{\Gamma \vdash T \leq V}$$

Transitivity elimination is essential, from the point of view of type-checking. A subtype checker uses deduction rules backward to generate subproblems from a subtype checking problem; this approach is not possible with the transitivity rule, which transforms the problem $\Gamma \vdash T \leq V$ in the infinite disjunction of all the ($\Gamma \vdash T \leq U$, $\Gamma \vdash U \leq V$) problems which

[1] Dipartimento di Informatica, Università di Pisa, Corso Italia 40, I-56125, Pisa, Italy, ghelli@di.unipi.it. This work was carried out with the partial support of E.E.C., Esprit Basic Research Action 6309 FIDE2 and of "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" of the Italian National Research Council under grant No.91.00877.PF69.

can be generated by choosing any type U. From the point of view of theoretical studies, the transitivity rule makes it impossible to reason about subtyping judgements via induction on the types involved in their proof, since not only are T and V in the premises not proper subterms of T and V in the result, but, moreover, there is no relation, a priori, between the size of U and that of T and V.

The fact that "transitivity elimination" does not hold has at least three important consequences:

- Implementation of type checking: the standard type-checking algorithm for $F_{\leq}$, which is complete for that system, is no longer complete when applied to the recursion-free fragment of $\mu F_{\leq}$. In other words, extending that algorithm to the recursive case does not mean just adding code to deal with recursive types, but means altering its behavior when dealing with non recursive types too. At present it is not clear how this can be accomplished, thought our results might give some suggestions.
- Language design: the only known proof of the existence of a minimum type for $F_{\leq}$, which is based on transitivity elimination, is no longer valid for $\mu F_{\leq}$. The existence of a minimum type is essential for the usability of the system as a basis for a programming language and is important for writing efficient type-checkers. We conjecture that $\mu F_{\leq}$ still enjoys that property, but we do not know how to prove it.
- Semantics: an interpretation of derivable $F_{\leq}$ judgements which is defined by induction on their derivation proof, is coherent when it depends only on the judgement and not on the proof. Coherence is a crucial property of interpretations. In [CuGhe] a set of "coherence equations" was defined, which allow us to reduce the problem of checking the coherence of any coercion-based model of $F_{\leq}$ to the validity of those equations in that model. With non-conservativity, satisfaction of these equations could be not sufficient to ensure even the coherence of the interpretation of $\mu F_{\leq}$ judgements not involving recursive types.

$F_{\leq}$ subtyping is known to be undecidable [Pie]. When an undecidable proof system is conservatively extended, its undecidability is inherited by the extended system, since otherwise the decision procedure for the "bigger" system could be used to decide provability in the "smaller" one. Hence, our result implies that undecidability of $F_{\leq}$ is not immediately inherited by $\mu F_{\leq}$, but has to be proved independently. Moreover in this paper we show that Pierce's proof of undecidability cannot be trivially rephrased in the context of $\mu F_{\leq}$.

However, as a partial conservativity result, we show that the set of subtyping judgements which can be expressed in $F_{\leq}$ but proved only in $\mu F_{\leq}$ is contained in a very peculiar class of judgements, the "diverging" judgements, i.e. those which make the standard subtype checking algorithm diverge. This result is important since it shows that a strict relation exists between the undecidability of $F_{\leq}$ and our non-conservativity result, and suggests a way to prove that recursive types are conservative over the decidable variants of $F_{\leq}$, like Kernel Fun (see [Ghe] and [GhePie]) and $F_{\leq}$ without Top (see [Ka]). Furthermore, it has been argued that the behavior of a type checking algorithm on diverging judgements is of no practical interest, since there is, in practice, no possibility that a real program contains such a judgement. This point of view is supported by the results in [Ghe] which describe the structure of the diverging judgements, which seems far more complex than what is actually used in practice. Adopting this point of view, our partial conservativity result means that, restricting the attention to "practically relevant" judgements, $\mu F_{\leq}$ is

conservative over $F_{\leq}$.

The paper is structured as follows. In Section 2 we give the preliminary definitions about $F_{\leq}$ and $\mu F_{\leq}$. In Section 3 we prove non conservativity of $\mu F_{\leq}$ over $F_{\leq}$. In Section 4 we characterize the set of provable or diverging judgements as the maximal relation enjoying some closure properties. This result is used in Section 5 to prove the partial conservativity result. All the sketched proofs are fully developed in the full paper.

2 Definitions

2.1 $F_{\leq}$

The syntax of $F_{\leq}$ subtyping judgements is defined as follows ($F_{\leq}$ terms and typing judgements have no relevance in this paper):

Types	T	::= t | Top | T→T | ∀t≤T. T
Environments	Γ	::= () | Γ, t≤T
Judgements	J	:= Γ ⊢ T ≤ T

$F_{\leq}$ types are either variables, Top, function types (T→T) or universally quantified types, used to give a type to polymorphic functions, where the type variable can be bounded. $F_{\leq}$ subtype rules are listed in Appendix A. We will often ignore function types in the following since, from the point of view of subtyping, they behave exactly like ∀ types where the quantified variable does not appear in the codomain. Variables are actually just names for their De Bruijn indexes, so that judgements and types are always considered modulo α conversion and all substitutions must be performed in a capture free way.

Due to transitivity, $F_{\leq}$ subtype rules are not a good tool for reasoning about non-derivability of judgements. This problem was solved in [CuGhe], by proving that any derivable judgement admits a normal form proof, i.e. a proof where transitivity is used only to prove judgements like Γ ⊢ t ≤ U as a consequence of Γ ⊢ t ≤ Γ(t) and Γ ⊢ Γ(t) ≤ U where Γ(t) is the bound of t in Γ (i.e., if Γ = (Γ', t≤T, Γ") then Γ(t)=$_{def}$ T). The existence of such a normal form proof for any subtyping judgement is what we called transitivity elimination in the introduction.

It is easy to show that an algorithm exists which, given any derivable subtyping judgement, rebuilds a normal form proof tree for that judgement. That algorithm can be described by the set of rewrite rules below. These rules transform a derivable judgement into the premises of the last subtyping rule applied in the normal form proof of that judgement.

(top)		Γ ⊢ T ≤ Top	▷	**true**
(varId)		Γ ⊢ u ≤ u	▷	**true**
(varTrans)	T≠u, T≠Top ⇒	Γ ⊢ u ≤ T	▷	Γ ⊢ Γ(u)[2] ≤ T
(∀dom)		Γ ⊢ ∀t≤T.U' ≤ ∀t≤T'.U	▷	Γ ⊢ T' ≤ T
(∀cod)		Γ ⊢ ∀t≤T.U' ≤ ∀t≤T'.U	▷	Γ, t≤T' ⊢ U' ≤ U[3]

The algorithm tries to build a normal-form proof tree starting from the conclusion and building (backward) all the branches by following all the possible rewriting chains generated by the conclusion. If some chain generates an irreducible judgement, different from **true**, then the algorithm fails. When all chains terminate with **true**, the set of these

[2] Variables in Γ(u) must be renamed to avoid capture; e.g., we could rename all variables in Γ⊢u≤T so that no variable name is used twice.

[3] We could write Γ ⊢ ∀t≤T.U' ≤ ∀t'≤T'.U ▷ Γ, t'≤T' ⊢ [t'/t]U' ≤ U to emphasize that t and t' can be different.

chains represents a proof tree for the original judgement[4], as depicted in Figure 1, and the algorithm terminates with success.

$$t \le Top,\ u \le t \to t \vdash u \le t \to Top$$
$$\triangleright (varTrans)$$
$$t \le Top,\ u \le t \to t \vdash t \to t \le t \to Top$$
$$\triangleright (\to dom) \qquad\qquad \triangleright (\to cod)$$
$$t \le Top,\ u \le t \to t \vdash t \le t \qquad\qquad t \le Top,\ u \le t \to t \vdash t \le Top$$
$$\triangleright (varId) \qquad\qquad \triangleright (top)$$
$$\textbf{true} \qquad\qquad \textbf{true}$$

$$\text{(Trans)}\ \frac{\dfrac{}{t \le Top,\ u \le t \to t \vdash u \le t \to t}\,(Var) \quad (\to)\dfrac{\dfrac{}{t \le Top,\ u \le t \to t \vdash t \le t}\,(Id) \qquad \dfrac{}{t \le Top,\ u \le t \to t \vdash t \le Top}\,(Top)}{t \le Top,\ u \le t \to t \vdash t \to t \le t \to Top}}{t \le Top,\ u \le t \to t \vdash u \le t \to Top}$$

Fig. 1: Correspondence between the chains generated by a judgement and its proof.

Proposition 2.1: The algorithm described by the five rules above is correct and complete with respect to $F_{\le}$ subtyping, i.e. it terminates with success on all and only the derivable judgements, while it may either fail or loop forever with non derivable judgements [CuGhe,Ghe]. The looping behavior cannot be avoided, since $F_{\le}$ subtyping is undecidable [Pie].

2.2 $\mu F_{\le}$

$\mu F_{\le}$ extends $F_{\le}$ by adding recursive types:

Types T ::= t | X | Top | T→T | ∀t≤T. T | μX.T→T | μX.∀t≤T. T

For clarity, we use two different families *t* and *X* for, respectively, bounded and recursive type variables. ∀ and μ bind the corresponding variables. We forbid recursive types whose body is either t, X, Top or $\mu X.T$. This is a minor restriction which simplifies some proofs without restricting the expressive power of the system. In fact a recursive type denotes the regular tree which constitutes its infinite unfolding. So, μX.Top, μX.t and μX.Y (X≠Y) can be forbidden, since they would denote just Top, t and Y. μX.X is forbidden, since it does not mean anything, and this is the only essential restriction. Finally, μX.μY.T can be forbidden since its unfolding is the same as μX.[X/Y]T [AmCa].

Different notions of type equality and subtyping can be defined for recursive types (see [AmCa] for an excellent discussion). The essential requirement is that a recursive type must be equal, or at least isomorphic, to any of its unfoldings. This requirement can be expressed by the following rule[5] (note the use of $\vdash_{\mu}$ for $\mu F_{\le}$ judgements):

---

[4] Actually the rebuilt tree is slightly different from the normal form proof, since a) the (exp) rule does not transform $\Gamma \vdash t \le T$ into the axiom $\Gamma \vdash t \le \Gamma(t)$ plus $\Gamma \vdash \Gamma(t) \le T$, but takes for granted the proof of the axiom and generates only $\Gamma \vdash \Gamma(t) \le T$; b) the rewrite rules prove $\Gamma \vdash T \le T$ by analyzing step by step the structure of T, applying identity only on atoms; these are just algorithmic optimizations which do not affect the correspondence between the two approaches.

[5] A less practical kind of recursive types can be obtained by not requiring that μX.T=[μX.T/X]T, but simply that for each recursive type there exist two functions, $fold_{(\mu X.T)}$:[μX.T/X]T→μX.T and $unfold_{(\mu X.T)}$:μX.T→[μX.T/X]T, such that fold(unfold(a))⊳a and unfold(fold(a))⊳a; this approach is adopted, for example, in Cardelli's Fsub system [Ca]. Our result does not apply to this approach.

(unfold) $\dfrac{\Gamma \vdash_\mu \mu X.T \quad \text{type}}{\Gamma \vdash_\mu \mu X.T = [\mu X.T/X]T}$

Since we are only interested in subtyping, and not in type equality, we will use the following two rules instead:

(unfold-l) $\dfrac{\Gamma \vdash_\mu [\mu X.T/X]T \le U}{\Gamma \vdash_\mu \mu X.T \le U}$ (unfold-r) $\dfrac{\Gamma \vdash_\mu U \le [\mu X.T/X]T}{\Gamma \vdash_\mu U \le \mu X.T}$

These are weak rules, which do not allow us to deduce all the subtyping judgements which could be proved using the Amadio-Cardelli system; for example you cannot prove that $\mu X.t \to t \to X$ is a subtype of $\mu X.t \to X$, even if the two types have the same infinite unfolding (see [AmCa]). However we will show that these rules are already powerful enough to extend $F_\le$ subtyping in a non-conservative way, i.e. to prove subtyping judgements which can be expressed but not proved in plain $F_\le$.

Using the backward notation, these two rules are written as follows:

$\Gamma \vdash_\mu \mu X.T \le U \ \triangleright\ \Gamma \vdash_\mu [\mu X.T/X]T \le U$ (unfold-l)
$\Gamma \vdash_\mu U \le \mu X.T \ \triangleright\ \Gamma \vdash_\mu U \le [\mu X.T/X]T$ (unfold-r)

Proposition 2.2: Adding the two rules above to the five rules for $F_\le$ subtyping we obtain a rule system for $\mu F_\le$ judgements which is sound but not complete.

The fact that the seven-rule system is sound can be easily verified, by showing that they actually build, in a backward fashion, a valid $\mu F_\le$ proof. The fact that they are not complete, even with respect to the weak recursive subtyping rules adopted, can be verified by observing that the judgement, provable in $\mu F_\le$ but not in $F_\le$, which we will exhibit in the next section is not proved by the seven-rule system. Non completeness of this set of rules is what we call "loss of transitivity elimination", since the only difference between the backward rules and the complete system is that the backward rules only use the limited form of transitivity formalized by the (varTrans) rule.

Soundness and completeness of the five rules means that we can use them to prove both derivability and non-derivability in $F_\le$, while the seven-rule set can be used only as an alternative way to prove derivability in $\mu F_\le$.

3 The Counterexample

In this section we will present a counterexample to the conservativity conjecture, i.e. an $F_\le$ judgement which cannot be derived in $F_\le$ but can be derived in $\mu F_\le$.

Notation: To improve readability, we will exploit the following conventions:

$\forall t.T$	stands for	$\forall t \le Top.T$
$-T$	stands for	$T \to Top$

And we will exploit the following derived rules:

($\forall$d)	Γ	$\vdash$	$\forall t.T' \le$	$\forall u \le U.T$	$\triangleright$	$\Gamma, u \le U$	$\vdash$	$[u/t]T'$	$\le$	T
(-)	Γ	$\vdash$	$-T \le$	$-U$	$\triangleright$	Γ	$\vdash$	U	$\le$	T
(var)	Γ	$\vdash$	$t \le$	Rename($\Gamma(t)$)	$\triangleright$	**true**				

(in the last rule Rename($\Gamma(t)$) stands for any α-variant of the bound of *t* in Γ).

Let $B = \forall u.-\forall v \le u.-u$. Then the following judgement cannot be derived in $F_\le$:

$t_0 \le B \vdash t_0 \le \forall t_1 \le t_0.-t_0$

This can be proved by showing that the complete $F_{\leq}$ subtyping algorithm described in the previous section diverges as follows:

let $B=\forall u.\text{-}\forall v{\leq}u.\text{-}u$:

$t_0{\leq}B$	$\vdash$	t_0	$\leq$	$\forall t_1{\leq}t_0.\text{-}t_0$	$\triangleright$(varTrans)
$t_0{\leq}B$	$\vdash$	$\forall t_1.\text{-}\forall t_2{\leq}t_1.\text{-}t_1$	$\leq$	$\forall t_1{\leq}t_0.\text{-}t_0$	$\triangleright(\forall)$
$t_0{\leq}B,t_1{\leq}t_0$	$\vdash$	$\text{-}\forall t_2{\leq}t_1.\text{-}t_1$	$\leq$	$\text{-}t_0$	$\triangleright$(-)
$t_0{\leq}B,t_1{\leq}t_0$	$\vdash$	t_0	$\leq$	$\forall t_2{\leq}t_1.\text{-}t_1$	

The judgement above is a special case (i=1) of the family below:

$$J_i = t_0{\leq}B,\ldots,t_{i+1}{\leq}t_i \ \vdash\ t_0 \ \leq\ \forall t_{i+2}{\leq}t_{i+1}.\text{-}t_{i+1}$$

For any *i* that judgement is transformed in the judgement J_{i+1}:

$t_0{\leq}B,\ldots,t_{i+1}{\leq}t_i$	$\vdash$	t_0	$\leq \forall t_{i+2}{\leq}t_{i+1}.\text{-}t_{i+1}$	$\triangleright$(varTrans)
$t_0{\leq}B,\ldots,t_{i+1}{\leq}t_i$	$\vdash$	$\forall t_{i+2}.\text{-}\forall t_{i+3}{\leq}t_{i+2}.\text{-}t_{i+2}$	$\leq \forall t_{i+2}{\leq}t_{i+1}.\text{-}t_{i+1}$	$\triangleright(\forall)$
$t_0{\leq}B,\ldots,t_{i+1}{\leq}t_i,t_{i+2}{\leq}t_{i+1}$	$\vdash$	$\text{-}\forall t_{i+3}{\leq}t_{i+2}.\text{-}t_{i+2}$	$\leq \text{-}t_{i+1}$	$\triangleright$(-)
$t_0{\leq}B,\ldots,t_{i+1}{\leq}t_i,t_{i+2}{\leq}t_{i+1}$	$\vdash$	t_{i+1}	$\leq \forall t_{i+3}{\leq}t_{i+2}.\text{-}t_{i+2}$	$\triangleright$(varTrans)
$t_0{\leq}B,\ldots,t_{i+1}{\leq}t_i,t_{i+2}{\leq}t_{i+1}$	$\vdash$	t_i	$\leq \forall t_{i+3}{\leq}t_{i+2}.\text{-}t_{i+2}$	$\triangleright$(varTrans)
…				$\triangleright$(varTrans)
$t_0{\leq}B,\ldots,t_{i+1}{\leq}t_i,t_{i+2}{\leq}t_{i+1}$	$\vdash$	t_0	$\leq \forall t_{i+3}{\leq}t_{i+2}.\text{-}t_{i+2}$	

This implies that the algorithm never terminates on the original judgement, hence, by completeness, that judgement cannot be proved in $F_{\leq}$[6].

We show now that in $\mu F_{\leq}$ there exists a (recursive) type T, such that:

(a) $t_0{\leq}B \vdash t_0 \leq T$
(b) $t_0{\leq}B \vdash T \leq \forall t_1{\leq}t_0.\text{-}t_0$

So that, by transitivity, the judgement above is derivable in $\mu F_{\leq}$; T is the following type:

$$T = \mu X.\forall t{\leq}X.\text{-}X.$$

We give the backward representation of a proof of the two judgements (a) and (b) above. When a $\forall$ judgement can be rewritten in two different ways we indicate with (1) the beginning of the rewriting chain of the first premise and with (2) the beginning of the rewriting chain of the second premise. Note that, while to prove non-derivability of the judgement above in $F_{\leq}$ we *needed* the backward representation, in this case, where we are only interested in the derivability of the judgement, we could write its proof in the usual tree form. We still resort to the backward representation to make it easier for the reader to compare the behavior of these two judgements with the behavior of the judgement above; the tree form of a proof of the judgement can be easily recovered.

(a)	$t_0{\leq}B$	$\vdash$	t_0	$\leq$	T		$\triangleright$(varTrans)
	$t_0{\leq}B$	$\vdash$	$\forall t_1.\text{-}\forall t_2{\leq}t_1.\text{-}t_1$	$\leq$	T		$\triangleright$(unfold-r) ($T=\mu X.\forall t{\leq}X.\text{-}X$)
	$t_0{\leq}B$	$\vdash$	$\forall t_1.\text{-}\forall t_2{\leq}t_1.\text{-}t_1$	$\leq$	$\forall t_1{\leq}T.\text{-}T$		$\triangleright(\forall)$
	$t_0{\leq}B,\ t_1{\leq}T$	$\vdash$	$\text{-}\forall t_2{\leq}t_1.\text{-}t_1$	$\leq$	$\text{-}T$		$\triangleright$(-)
	$t_0{\leq}B,\ t_1{\leq}T$	$\vdash$	T	$\leq$	$\forall t_2{\leq}t_1.\text{-}t_1$		$\triangleright$(unfold-l)
	$t_0{\leq}B,\ t_1{\leq}T$	$\vdash$	$\forall t_2{\leq}T.\text{-}T$	$\leq$	$\forall t_2{\leq}t_1.\text{-}t_1$		$\triangleright(\forall\text{dom})$:**1**; $\triangleright(\forall\text{cod})$:**2**
	1 $t_0{\leq}B,\ t_1{\leq}T$	$\vdash$	t_1	$\leq$	T		$\triangleright$(var) **true**

[6] This underivable judgement was introduced in [Ghe]; before its discovery, it was widely believed that the algorithm described in the previous section was terminating, and so that $F_{\leq}$ subtyping was decidable. A variant of this judgement has then been used by Pierce to encode two-counter Turing machines as $F_{\leq}$ subtyping judgements, proving undecidability of $F_{\leq}$ subtyping [Pie].

	2	$t_0 \le B, t_1 \le T, t_2 \le t_1$	$\vdash$	$-T$	$\le$	$-t_1$	$\triangleright$(-)
		$t_0 \le B, t_1 \le T, t_2 \le t_1$	$\vdash$	t_1	$\le$	T	$\triangleright$(var) **true**
(b)	$t_0 \le B$		$\vdash$	T	$\le$	$\forall t_1 \le t_0.-t_0$	$\triangleright$(unfold-l)
	$t_0 \le B$		$\vdash$	$\forall t_1 \le T.-T$	$\le$	$\forall t_1 \le t_0.-t_0$	$\triangleright$(∀dom):**1**; $\triangleright$(∀cod):**2**
	1	$t_0 \le B$	$\vdash$	t_0	$\le$	T	$\triangleright$* **true** (see (a))
	2	$t_0 \le B, t_1 \le t_0$	$\vdash$	$-T$	$\le$	$-t_0$	$\triangleright$(-)
		$t_0 \le B, t_1 \le t_0$	$\vdash$	t_0	$\le$	T	$\triangleright$* **true** (see (a), t_1 plays no role)

As already stated, the non conservativity of $\mu F_{\le}$ w.r.t. $F_{\le}$ implies that the undecidability of the second system does not necessarily extend to the first one. Moreover the judgement that we have shown to become derivable in $\mu F_{\le}$ is exactly that judgement whose non derivability has been used by Pierce to show undecidability of $F_{\le}$ subtyping. This means that it is seems rather difficult to adapt Pierce's proof to show $\mu F_{\le}$ undecidability.

It could even be conceivable that adding recursive types to $F_{\le}$ types makes its type-checking problem easier, in the same way as admitting rational infinite terms as solutions for the unification problem makes its solution simpler, allowing us to avoid the occur check. However, this optimistic hypothesis is not convincing, since we do not see, in practice, how a complete algorithm for $\mu F_{\le}$ subtyping could be designed.

Our counterexample hints at a possible direction to explore in order to complete the algorithm. Since the structure of the recursive type T "reflects" the structure of the looping behavior of the judgement with respect to the standard type-checking algorithm (an alternation of ∀ and - steps), we could look for an algorithm which tries to build an intermediate, possibly recursive, type whose structure is derived from the behavior of the standard algorithm.

4 The Compatibility Relation Defined by Diverging Judgements

We want to prove that the effects of our non-conservativity result are limited to the "pathological" set of the diverging $F_{\le}$ judgements, i.e. to those judgements which make the $F_{\le}$ subtype checking algorithm diverge.

To prove this partial conservativity result we first need some results about the nature of diverging judgements. To this aim, in this section we study the point of view that, when the subtype-checking algorithm is not able to prove in a finite time that T is not a subtype of U, this means that T is somehow "compatible" with U, even if it is not exactly a subtype of U, and we will study the properties of this compatibility relation.

Compatibility will be defined by negation, as the complement of the notion of being "provably not a subtype". We first model the failing runs of the subtyping algorithm using the proof system below.

$$(t/Top\not\le)\quad \Gamma \vdash Top \not\le t \qquad (Top/\forall\not\le)\quad \Gamma \vdash Top \not\le \forall t \le T.U$$

$$(\forall/t\not\le)\quad \Gamma \vdash \forall t \le T.U \not\le t \qquad (VarTrans\not\le)\quad \frac{\Gamma \vdash \Gamma(t) \not\le T \quad T \ne t,\ T \ne Top}{\Gamma \vdash t \not\le T}$$

$$(\forall r\not\le)\quad \frac{\Gamma, t \le T \vdash U \not\le U'}{\Gamma \vdash \forall t \le T'.U \not\le \forall t \le T.U'} \qquad (\forall l\not\le)\quad \frac{\Gamma \vdash T \not\le T'}{\Gamma \vdash \forall t \le T'.U \not\le \forall t \le T.U'}$$

Proposition 4.1: $\Gamma \vdash T \not\le U$ if and only if the standard type-checking algorithm fails (in a finite time) on the judgement $\Gamma \vdash T \le U$.

Proof hint : By induction, on the number of steps executed by the algorithm in one

direction, and on the size of the proof of $\Gamma \vdash T \not\leq U$ in the other one.

In this section we will always consider all the relations written as $\Gamma \vdash T \mathcal{R} U$ as ranging on the universe J of well-formed judgements:

$$J = \{ \Gamma, T, U \mid \Gamma \vdash T \text{ type}, \Gamma \vdash U \text{ type}\}$$

Hereafter, $\Gamma \vdash T \not\leq U$ denotes the intersection of the relation defined by the rules above with J; note that it would be very easy to modify the proof system of $\not\leq$ so that it only produces such judgements.

For the sake of compactness of notation, we define a function $\mathcal{S}(T)$ ("shape of T") from $F_{\leq}$ types to the three element linearly ordered set $\{t < \forall < Top\}$ as follows:

for any type variable u:	$\mathcal{S}(u)$	$= t$
for any t,T,U:	$\mathcal{S}(\forall t \leq T.U)$	$= \forall$
	$\mathcal{S}(Top)$	$= Top$

Then we collect the three axioms of $\not\leq$ into one:

(Shape$\not\leq$) $\quad \mathcal{S}(T) > \mathcal{S}(U) \Rightarrow \Gamma \vdash T \not\leq U$

The compatibility relation can be now formally defined.

Notation: For any relation denoted as $\Gamma \vdash T \mathcal{R} U$:

$$\Gamma \nvdash T \mathcal{R} U \Leftrightarrow_{def} \langle \Gamma,T,U\rangle \in J \text{ and not } \Gamma \vdash T \mathcal{R} U.$$

(Note that $\nvdash$ is just a classical negation, so that $(\text{not}(\Gamma \nvdash T \mathcal{R} U)) \Leftrightarrow \Gamma \vdash T \mathcal{R} U$).

Definition: $\Gamma \vdash T \sqsubseteq U$ (read: T is compatible with U w.r.t. Γ) $\Leftrightarrow_{def} \Gamma \nvdash T \not\leq U$

By proposition 4.1, $\Gamma \vdash T \sqsubseteq U$ means that trying to prove $\Gamma \vdash T \leq U$ results either in success or looping. We call the $\sqsubseteq$ relation "compatibility", since $\Gamma \vdash T \sqsubseteq V$ means that V is so near to being a supertype of T that the standard algorithm is not able to prove the opposite.

To study the compatibility relation, we first need a "positive" connotation for it. To this aim, we first observe that compatibility shares with subtyping a set of "compatibility properties".

Definition: A relation $\mathcal{R}$ on J is *compatible with subtyping* (compatible, for short) if:

($\forall\mathcal{R}$)	$\Gamma \vdash \forall t \leq T'.U \mathcal{R} \forall t \leq T.U'$	$\Leftrightarrow \Gamma \vdash T \mathcal{R} T'$ and $\Gamma, t \leq T \vdash U \mathcal{R} U'$
(VarTrans$\mathcal{R}$) $U \neq Top, U \neq t$	$\Rightarrow \Gamma \vdash t \mathcal{R} U$	$\Leftrightarrow \Gamma \vdash \Gamma(t) \mathcal{R} U$
(Top$\mathcal{R}$)		$\Gamma \vdash T \mathcal{R} Top$
(Id$\mathcal{R}$)		$\Gamma \vdash t \mathcal{R} t$
(Shape$\mathcal{R}$)	$\Gamma \vdash T \mathcal{R} U$	$\Rightarrow \mathcal{S}(T) \leq \mathcal{S}(U)$

The compatibility properties give us a different way of characterizing $\leq$ and $\sqsubseteq$: we will prove that they are respectively the minimum and the maximum compatible relations.

First, we show that any compatible relation has no intersection with the relation $\not\leq$.

Proposition 4.2: If $\mathcal{R}$ is compatible and $\Gamma \vdash T \not\leq U$, then $\Gamma \nvdash T \mathcal{R} U$.

Proof hint : By induction on the proof of $\Gamma \vdash T \not\leq U$, and by cases on the last rule applied. Essentially, any proof of $\Gamma \vdash T \not\leq U$ corresponds to a chain of applications of ($\forall\mathcal{R}$) and (VarTrans$\mathcal{R}$) which transform $\Gamma \vdash T \not\leq U$ into a contradiction to (Shape$\mathcal{R}$).

Now the "positive characterization" of the compatibility relation can be given; the

following proposition is all that is needed in the next section to prove partial conservativity.

Proposition 4.3: $\sqsubseteq$ is the maximum compatible relation on J.

Proof hint : First we prove that $\sqsubseteq$ is compatible. Since $\sqsubseteq$ is defined as the complement of $\not\sqsubseteq$, we first negate both sides of all the properties, to express them in terms of $\not\sqsubseteq$:

($\forall\sqsubseteq$)	$\Gamma \vdash \forall t{\leq}T'.U \not\sqsubseteq \forall t{\leq}T.U'$	$\Leftrightarrow \Gamma \vdash T \not\sqsubseteq T'$ or $\Gamma, t{\leq}T \vdash U \not\sqsubseteq U'$
(VarTrans$\sqsubseteq$) $U{\neq}Top, U{\neq}t \Rightarrow$	$\Gamma \vdash t \not\sqsubseteq U$	$\Leftrightarrow \Gamma \vdash \Gamma(t) \not\sqsubseteq U$
(Top$\sqsubseteq$)		$\Gamma \nvdash T \not\sqsubseteq Top$
(Id$\sqsubseteq$)		$\Gamma \nvdash t \not\sqsubseteq t$
(Shape$\sqsubseteq$)	$\Gamma \vdash T \not\sqsubseteq U$	$\Leftarrow \mathcal{S}(T) > \mathcal{S}(U)$

Maximality of $\sqsubseteq$ is a corollary of proposition 4.2: since, for any compatible $\mathcal{R}$,

$\Gamma \vdash T \not\sqsubseteq U \Rightarrow \Gamma \nvdash T \mathcal{R} U$, then $\Gamma \vdash T \mathcal{R} U \Rightarrow \Gamma \nvdash T \not\sqsubseteq U$, i.e. $\Gamma \vdash T \sqsubseteq U$.

Proposition 4.4: $\leq$ is the minimum compatible relation on J.

Proof hint : Compatibility and minimality of $\leq$ are consequences of completeness and soundness of the subtype-checking algorithm presented in Section 2.

The fact that there are at least two different compatible relations could be surprising, since the compatibility conditions are rather restrictive. They contain both positive conditions, (Top) and (Id), which impose that something is in the relation, negative conditions (Shape) imposing that something is not in the relation, and bidirectional closure conditions, ($\forall$) and (VarTrans), which allow us to deduce both the presence and the absence of something from the presence and the absence of something else. Moreover any judgement matches at least one of the five conditions, so that it seems that you can use those conditions to decide whether any judgement belongs to $\mathcal{R}$. Obviously, this is not the case, and the problem is that trying to use consistency conditions to decide whether a pair of types is in $\mathcal{R}$ can result in an infinite loop. On the other hand, in the decidable variants of $F_{\leq}$, the repeated application of the compatibility conditions always produces a negative or a positive axiom, so that in those systems subtyping is really the only compatible relation. So, the existence of a full range of different compatible relations is a feature of $F_{\leq}$ which is strictly related to its undecidability.

Before going back to the main stream of the paper let us pause to show that the compatible relation is similar to $\leq$ also by enjoying the properties of transitivity, narrowing and irreflexivity (corollary 4.9).

We first prove that any compatible relation is irreflexive.

Definition: If t is defined in a context Γ, the definition level of t in Γ, written $\mathcal{L}(t,\Gamma)$, is the number of variable definitions in Γ which strictly precede the definition of t:

$\mathcal{L}(t, (t{\leq}A, \Gamma)) = 0$

$\mathcal{L}(t, (u{\leq}A, \Gamma)) = \mathcal{L}(t, \Gamma)+1 \quad (t{\neq}u)$

The definition level of a type T well formed in Γ, written $\mathcal{L}(T,\Gamma)$, is the maximum definition level of the variables which are free in T, and is 0 if no variable is free in T.

Fact 4.5: In any well-formed context, $\mathcal{L}(\Gamma(t),\Gamma){<}\mathcal{L}(t,\Gamma)$, since all the free variables in $\Gamma(t)$ have to be defined before t.

Lemma 4.6: Let $\mathcal{R}$ be a compatible relation. Then $\Gamma \nvdash \Gamma(t)\ \mathcal{R}\ t$.

Proof hint : By (Shape$\mathcal{R}$), $\Gamma \vdash \Gamma(t)\ \mathcal{R}\ t$ would imply $\Gamma(t)=u$ for a certain u; by Fact 4.5,

this implies $\mathcal{L}(u,\Gamma)\leq\mathcal{L}(t,\Gamma)$; so we find a contradiction by proving, by induction on $\mathcal{L}(u,\Gamma)$, that: $(\ \Gamma\vdash u\ \mathcal{R}\ t$ and $u\neq t\ )\ \Rightarrow\ (\ \mathcal{L}(u,\Gamma) > \mathcal{L}(t,\Gamma)\ )$.

Proposition 4.7: Any compatible $\mathcal{R}$ is irreflexive: $\Gamma\vdash T\ \mathcal{R}\ U$ and $\Gamma\vdash U\ \mathcal{R}\ T\ \Rightarrow T=U$.

Proof : By induction on the size of T and U. By (Shape$\mathcal{R}$), $\Gamma\vdash T\ \mathcal{R}\ U$ and $\Gamma\vdash U\ \mathcal{R}\ T$ implies that $\mathcal{S}(T)=\mathcal{S}(U)$. Case $\mathcal{S}(T)=t$ is managed by 4.6, the others are routine.

Now the transitivity of $\sqsubseteq$ can be proved. Since we cannot reason by induction on the $\sqsubseteq$ relation, we will reason by maximality, by proving that the transitive closure of $\sqsubseteq$ is a compatible relation which contains $\sqsubseteq$.

Notation: If $\mathcal{R}$ is a relation on J, its lifting $\mathcal{R}_\Gamma$ to well formed environments is defined inductively as follows:

$$() \ \mathcal{R}_\Gamma\ ()$$
$$\Gamma, t\leq T\ \mathcal{R}_\Gamma\ \Delta, t\leq U\ \Leftrightarrow_{def}\ \Gamma\ \mathcal{R}_\Gamma\ \Delta \text{ and } \Gamma\vdash T\ \mathcal{R}\ U$$

Now we define the transitive closure $\sqsubseteq^+$ of the $\sqsubseteq$ relation; condition (Narrow+) is strictly needed, since narrowing is essentially another aspect of transitivity, as shown in the proof of Proposition 4.8. Conditions ($\forall$+) and (VarTrans+) are added to the definition of $\sqsubseteq^+$ to simplify the same proof.

Definition: $\Gamma\vdash T\sqsubseteq^+ V$ is the relation defined by the following five deduction rules:

($\sqsubseteq$)	$\Gamma\vdash T\sqsubseteq V$	$\Rightarrow$	$\Gamma\vdash T\sqsubseteq^+ V$
(Trans+)	$\Gamma\vdash T\sqsubseteq^+ U$ and $\Gamma\vdash U\sqsubseteq^+ V$	$\Rightarrow$	$\Gamma\vdash T\sqsubseteq^+ V$
(Narrow+)	$\Gamma\vdash T\sqsubseteq^+ U$ and $\Delta\sqsubseteq^+\Gamma$	$\Rightarrow$	$\Delta\vdash T\sqsubseteq^+ U$
($\forall$+)	$\Gamma\vdash T\sqsubseteq^+ T'$ and $\Gamma, t\leq T\vdash U\sqsubseteq^+ U'$	$\Rightarrow$	$\Gamma\vdash \forall t\leq T'.U\sqsubseteq^+ \forall t\leq T.U'$
(VarTrans+)	$U\neq$Top, $U\neq t$, $\Gamma\vdash\Gamma(t)\sqsubseteq^+ U$	$\Rightarrow$	$\Gamma\vdash t\sqsubseteq^+ U$

Proposition 4.8: $\sqsubseteq^+$ is equal to $\sqsubseteq$.

Proof : $\sqsubseteq^+\supseteq\sqsubseteq$ is immediate by condition ($\sqsubseteq$) above. We show that $\sqsubseteq^+$ is compatible; the thesis follows by maximality of $\sqsubseteq$ among compatible relations. We show the typical case, ($\forall\sqsubseteq^+$): $\Gamma\vdash\forall t\leq T'.U\sqsubseteq^+\forall t\leq T.U'\ \Leftrightarrow\ \Gamma\vdash T\sqsubseteq^+ T'$ and $\Gamma, t\leq T\vdash U\sqsubseteq^+ U'$

$\Leftarrow$: by ($\forall$+)

$\Rightarrow$: By induction on the size of the proof that $\Gamma\vdash\forall t\leq T'.U\sqsubseteq^+\forall t\leq T.U'$ and by cases on the last rule applied. ($\sqsubseteq$): by ($\forall\sqsubseteq$). ($\forall$+): immediate. (VarTrans+): impossible.

(Narrow+): exists Δ: $\Gamma\sqsubseteq^+\Delta$ and $\Delta\vdash\forall t\leq T'.U\sqsubseteq^+\forall t\leq T.U'$
$\Rightarrow$(Ind.) $\Delta\vdash T\sqsubseteq^+ T'$ and $\Delta, t\leq T\vdash U\sqsubseteq^+ U'$
$\Rightarrow$(Narrow+) $\Gamma\vdash T\sqsubseteq^+ T'$ and $\Gamma, t\leq T\vdash U\sqsubseteq^+ U'$.

(Trans+): by (Shape$\sqsubseteq^+$) the intermediate type has the shape $\forall t\leq V.W$:
$\Gamma\vdash\forall t\leq T'.U\sqsubseteq^+\forall t\leq V.W$, $\Gamma\vdash\forall t\leq V.W\sqsubseteq^+\forall t\leq T.U'$
$\Rightarrow$(Ind.) $\Gamma\vdash V\sqsubseteq^+ T'$ and $\Gamma, t\leq V\vdash U\sqsubseteq^+ W$,
$\Gamma\vdash T\sqsubseteq^+ V$ and $\Gamma, t\leq T\vdash W\sqsubseteq^+ U'$
$\Rightarrow$(Narrow+) $\Gamma\vdash V\sqsubseteq^+ T'$ and $\Gamma, t\leq T\vdash U\sqsubseteq^+ W$,
$\Gamma\vdash T\sqsubseteq^+ V$ and $\Gamma, t\leq T\vdash W\sqsubseteq^+ U'$
$\Rightarrow$(Trans+) $\Gamma\vdash T\sqsubseteq^+ T'$ and $\Gamma, t\leq T\vdash U\sqsubseteq^+ U'$.

Corollary 4.9: $\sqsubseteq$ enjoys both transitivity and narrowing, i.e.:

$$\Gamma\vdash T\sqsubseteq U \text{ and } \Gamma\vdash U\sqsubseteq V\ \Rightarrow\ \Gamma\vdash T\sqsubseteq V$$
$$\Gamma\vdash T\sqsubseteq U \text{ and } \Delta\sqsubseteq\Gamma\ \Rightarrow\ \Delta\vdash T\sqsubseteq U$$

This study of compatible relations leaves two interesting problems open:

- Does compatibility imply transitivity?
- Is there some other compatible relation apart from $\leq$ and $\sqsubseteq$? Is there some other interesting compatible relation apart from $\leq$ and $\sqsubseteq$?

We have no answer for the first question, while the second will be answered in the next section.

5 Conservativity of $\mu F_{\leq}$ Over Terminating Judgements

In Section 3 we showed that $\mu F_{\leq}$ is not conservative over $F_{\leq}$, by exhibiting a non provable $F_{\leq}$ judgement which is provable in $\mu F_{\leq}$. That judgement is "divergent", which means that it makes the standard $F_{\leq}$ type-checking algorithm diverge. In this section we generalize that observation, by showing that every "finitely failing" judgement, i.e. which makes the standard $F_{\leq}$ type-checking algorithm fail, is not provable in $\mu F_{\leq}$, so that non conservativity is restricted to the "pathological" set of the diverging judgements.

We show this by proving that $\mu F_{\leq}$ subtyping is still a compatible relation, so that it cannot relate more types than the maximum compatible relation "$\sqsubseteq$".

Definition: μJ is the set of triples which form well-formed $\mu F_{\leq}$ judgements:

$$\mu J = \{ \Gamma, T, U \mid \Gamma \vdash_{\mu} T \text{ type}, \Gamma \vdash_{\mu} U \text{ type}\}$$

Definition: A relation $\mathcal{R}$ on μJ is μ-compatible when:

$(\forall\mathcal{R})$	$\Gamma \vdash \forall t{\leq}T.U \mathcal{R} \forall t{\leq}T.U'$	$\Leftrightarrow \Gamma \vdash T \mathcal{R} T'$ and $\Gamma, t{\leq}T \vdash U \mathcal{R} U'$
(VarTrans$\mathcal{R}$) $U{\neq}Top, U{\neq}t \Rightarrow$	$\Gamma \vdash t \mathcal{R} U$	$\Leftrightarrow \Gamma \vdash \Gamma(t) \mathcal{R} U$
(Top$\mathcal{R}$)		$\Gamma \vdash T \mathcal{R}$ Top
(Id$\mathcal{R}$)		$\Gamma \vdash t \mathcal{R} t$
(Shape$\mathcal{R}$)	$\Gamma \vdash T \mathcal{R} U$	$\Rightarrow \mathcal{S}(T) \leq \mathcal{S}(U)$
($\mu\mathcal{R}l$)	$\Gamma \vdash \mu X.T \mathcal{R} U$	$\Leftrightarrow \Gamma \vdash [\mu X.T/X]T \mathcal{R} U$
($\mu\mathcal{R}r$)	$\Gamma \vdash T \mathcal{R} \mu X.U$	$\Leftrightarrow \Gamma \vdash T \mathcal{R} [\mu X.U/X]U$

Where, in condition (Shape$\mathcal{R}$), $\mathcal{S}(\mu X.T) =_{def} \mathcal{S}(T)$

Proposition 5.1: The relation $\Gamma \vdash_{\mu} T \leq U$ is μ-compatible.

Proof : See Appendix B.

Lemma 5.2: Any μ-compatible relation is compatible when restricted to J.

Proof : Easy (see the full paper).

The main theorem now follows immediately.

Theorem 5.3: If $\Gamma \vdash_{\mu} T \leq U$ and Γ, T and U do not contain μ, then $\Gamma \vdash T \sqsubseteq U$

Proof : By proposition 5.1, the $\leq_{\mu}$ relation is μ-compatible; by lemma 5.2, its restriction to $F_{\leq}$ is compatible. The thesis follows from the maximality of $\sqsubseteq$ among the compatible relations (proposition 4.3).

We have proved that the restriction of $\leq_{\mu}$ to J is strictly bigger than $F_{\leq}$ subtyping but is contained in $\sqsubseteq$. We can now prove that $\leq_{\mu}$ restricted to J is *strictly* contained in $\sqsubseteq$.

Lemma 5.4: The $\sqsubseteq$ relation is co-R.E. but is not R.E.

Proof : See Appendix B.

Proposition 5.5: $\mu F_{\leq}$ subtyping restricted to J is R.E.

Proof : All the rules of $\mu F_{\leq}$ subtyping are effective, so that they can be used to enumerate

all the provable $\mu F_{\leq}$ subtyping judgements; J is a decidable subset of μJ.

Corollary: $\leq_{\mu}$ restricted to J is *strictly* contained in $\sqsubseteq$.

Hence, $\leq_{\mu}$ restricted to J defines a third compatible relation, intermediate between $F_{\leq}$ subtyping and $\sqsubseteq$, and different from both of them.

This fact implies that there exists a looping judgement which is not provable in $\mu F_{\leq}$. So, if we knew such a judgement, we could try to use it to rephrase Pierce's proof of undecidability for $\mu F_{\leq}$. The problem is recognizing that a looping judgement is not provable in $\mu F_{\leq}$. In fact, the only technique we know to prove that a judgement is not provable in $\mu F_{\leq}$ is to apply backward the deduction rules to it to reduce it to a judgement violating the (Shape≤) condition. However we have proved that this process never ends when is applied to a looping judgement (see proposition 4.3).

Note that compatibility of $\leq_{\mu}$ implies that, for every proof of $\Gamma \vdash_{\mu} T \leq U$, where the last rule used is transitivity, a proof exists for $\Gamma \vdash_{\mu} T \leq U$ where the last rule used is either (VarTrans) or a rule which is not transitivity; the key property, to prove this fact, is $(\forall \leq_{\mu})$. This sounds like a form of transitivity elimination. Actually, this fact only implies that, for any *n*, we can find a proof of $\Gamma \vdash_{\mu} T \leq U$ where full transitivity is not used in any of the last *n* steps, but does not imply that we can find a proof where full transitivity is *never* used. The point is that, intuitively, it is possible that whenever we remove a transitivity instance from the bottom of the proof, we add some new transitivity instance to the higher levels, so that the process of transitivity elimination never ends.

6 Conclusion and future work

We have proved that recursive types are not a conservative extension for the $F_{\leq}$ system. Moreover we have shown that in the extended system $\mu F_{\leq}$ transitivity elimination does not hold, i.e. it is not possible to substitute the general transitivity rule with the restricted (VarTrans) version. This implies that a rich set of proofs of properties of $F_{\leq}$, which is based on transitivity elimination, is not valid for $\mu F_{\leq}$ (see, e.g., [CuGhe], [GheTh], [CaMaMiSce], [CuGhe2], [BrLo]). We have shown that the basic diverging judgement of $F_{\leq}$ is provable in $\mu F_{\leq}$, thus showing that Pierce's proof of undecidability of $F_{\leq}$ subtyping is not valid for $\mu F_{\leq}$ ([Pie]).

Non conservativity has serious practical consequences: it means that the standard type checking algorithm for $F_{\leq}$ is no more complete with respect to $\mu F_{\leq}$ judgements, even for judgements not containing recursive types.

As a positive counterpart to this set of negative results, we have shown that the difference between $F_{\leq}$ and $\mu F_{\leq}$ is restricted to the set of the diverging judgements. This means that the standard $F_{\leq}$ type-checking algorithm is at least sound, even if not complete, when used to check $\mu F_{\leq}$ subtyping on $F_{\leq}$ judgements, since it does not give any answer on those judgements whose provability is different in the two systems. Furthermore, the standard type-checking algorithm for $F_{\leq}$ "goes wrong", when applied to μ-free $\mu F_{\leq}$ judgements, exactly in the same cases when it goes wrong with respect to $F_{\leq}$, i.e. on the diverging judgements. The only difference is that, in the case of $F_{\leq}$, the algorithm "goes wrong" by looping in some cases when it should fail, while, with $\mu F_{\leq}$, it loops on a set of judgements containing both provable and not provable ones. But the essential point is that those judgements have no serious possibility of appearing in a real program, so that a looping behavior of the algorithm in those cases may be tolerated. However, the actual impact of our result on the use of $F_{\leq}$ as a basis for language design should be more

seriously taken into consideration.

Finally, conservativity over terminating judgements shows that non-conservativity of recursion is connected to the problem of undecidability of subtyping, and gives a technique to prove that recursion is conservative over the decidable variants of $F_{\leq}$.

We chose a very weak form of recursive types for this study. This implies that the non conservativity result has a wide applicability, since it holds for any notion of recursive types which is stronger than the one adopted. On the other hand, for the same reason, our partial conservativity result has a more limited applicability, since it immediately holds only for notions of recursive types which are weaker than the one adopted. So our next step should be the definition of a strong notion of recursive types, generalizing the [AmCa] approach to $F_{\leq}$, to verify whether the partial conservativity result can be extended to that system.

Acknowledgements

The counterexample was obtained by slightly modifying a judgement produced by Roberto Bergamini whose suggestion sparked off this work.

This work was carried out in the framework of the Galileo project, aimed at the design of a strongly typed object-oriented database language, and lead by Prof. A. Albano at the University of Pisa.

References

[AmCa] R. Amadio, L. Cardelli, *Subtyping Recursive Types*, DEC SRC Research Report 62, short version in Proc. of the 18th ACM Symposium on Principles of Programming Languages, 104-118, 1991.

[BrLo] K. Bruce, G. Longo, A Modest Model of Records, Inheritance and Bounded Quantification, *Information and Computation 87 (1/2)*, 196-240, 1990.

[Ca] L. Cardelli, *Fsub: the System*, note, 1991.

[CaLo] L. Cardelli, G. Longo, *A Semantic Basis for Quest*, DEC SRC Research Report 55, short version in Proc. Conf. on Lisp and Functional Programming, Nice, 1990.

[CaMaMiSce] L. Cardelli, S. Martini, J. Mitchell, A. Scedrov, An Extension of System F with Subtyping, *Proc. Conference on Theoretical Aspects of Computer Software*, Sendai, Japan, Springer-Verlag, Berlin, LNCS 526, 1991.

[CaWe] L. Cardelli, P. Wegner, On Understanding Types, Data Abstraction and Polymorphism, *ACM Computing Surveys 17 (4)*, 471-522, 1985.

[CuGhe] P.-L. Curien, G. Ghelli, Coherence of Subsumption in $F_{\leq}$, Minimum Typing and Type Checking, *Mathematical Structures in Computer Science 2(1)*, 55-91, 1992.

[CuGhe2] P.-L. Curien, G. Ghelli, Subtyping + Extensionality: Confluence of $\beta\eta top_{\leq}$ in $F_{\leq}$, extended abstract, *Proc. Conference on Theoretical Aspects of Computer Software*, Sendai, Japan, Springer-Verlag, Berlin, LNCS 526, 731-749, 1991.

[Ghe] G. Ghelli, *Divergence of $F_{\leq}$ Type-checking*, note, 1991.

[GhePie] G. Ghelli, B. Pierce, *Bounded Existentials and Minimal Typing*, note, 1992.

[GheTh] G. Ghelli, *Proof-theoretic Studies about a Minimal Type System Integrating Inclusion and Parametric Polymorphism*, PhD Thesis, TD-6/90, Univ. of Pisa, 1990.

[Ka] D. Katiyar, S. Sankar, Completely bounded quantification is decidable, Proc. of the *ACM SIGPLAN Workshop on ML and its Applications*, 1992.

[Pie] B. Pierce, Bounded Quantification is Undecidable, *Proc. of the 19th ACM Symposium on Principles of Programming Languages*, 305-315, 1992.

Appendix A: $\mu F_{\leq}$.

Environments (sequences whose individual components have the form x:T or t≤T)

($\varnothing$env) $\quad$ () env

(≤env) $\quad \dfrac{\Gamma \text{ env} \quad \Gamma \vdash T \text{ type} \quad t \notin \Gamma}{\Gamma,\ t \leq T \text{ env}}$ [7] $\qquad$ (: env) $\quad \dfrac{\Gamma \text{ env} \quad \Gamma \vdash T \text{ type} \quad x \notin \Gamma}{\Gamma,\ x:T \text{ env}}$

Types

(VarForm) $\quad \dfrac{\Gamma, t \leq T, \Gamma' \text{ env}}{\Gamma, t \leq T, \Gamma' \vdash t \text{ type}}$ $\qquad$ (TopForm) $\quad \dfrac{\Gamma \text{ env}}{\Gamma \vdash \text{Top type}}$

(→ Form) $\quad \dfrac{\Gamma \vdash T \text{ type} \quad \Gamma \vdash U \text{ type}}{\Gamma \vdash T \to U \text{ type}}$ $\qquad$ ($\forall$ Form) $\quad \dfrac{\Gamma, t \leq T \vdash U \text{ type}}{\Gamma \vdash \forall t \leq T.\ U \text{ type}}$

(μForm) $\quad \dfrac{\Gamma \vdash [\text{Top}/X]U \text{ type}}{\Gamma \vdash \mu X.U \text{ type}}$

Subtypes

(Var≤) $\quad \dfrac{\Gamma, t \leq T, \Gamma' \text{ env}}{\Gamma, t \leq T, \Gamma' \vdash t \leq T}$ $\qquad$ (Top≤) $\quad \dfrac{\Gamma \vdash T \text{ type}}{\Gamma \vdash T \leq \text{Top}}$

(→≤) $\quad \dfrac{\Gamma \vdash T \leq T' \quad \Gamma \vdash U \leq U'}{\Gamma \vdash T' \to U \leq T \to U'}$ $\qquad$ ($\forall$≤) $\quad \dfrac{\Gamma \vdash T \leq T' \quad \Gamma, t \leq T \vdash U \leq U'}{\Gamma \vdash \forall t \leq T'.U \leq \forall t \leq T.U'}$

(Id≤) $\quad \dfrac{\Gamma \vdash T \text{ type}}{\Gamma \vdash T \leq T}$ $\qquad$ (Trans≤) $\quad \dfrac{\Gamma \vdash T \leq U \quad \Gamma \vdash U \leq V}{\Gamma \vdash T \leq V}$

Recursive types

(unfold-l) $\quad \dfrac{\Gamma \vdash [\mu X.T/X]T \leq U}{\Gamma \vdash \mu X.T \leq U}$ $\qquad$ (unfold-r) $\quad \dfrac{\Gamma \vdash U \leq [\mu X.T/X]T}{\Gamma \vdash U \leq \mu X.T}$

Algorithmic subtype rules: drop (Trans≤) and add:

(VarTrans≤) $\quad \dfrac{\Gamma, t \leq T, \Gamma' \vdash T \leq U \quad U \neq \text{Top}, U \neq t}{\Gamma, t \leq T, \Gamma' \vdash t \leq U}$

Appendix B: Proofs of propositions 5.1 and 5.4.

Proposition 5.1: The relation $\Gamma \vdash_{\mu} T \leq U$ is μ-compatible.

Proof : We have to prove that:

($\forall$≤)	$\Gamma \vdash \forall t \leq T'.U \leq \forall t \leq T.U'$	$\Leftrightarrow \Gamma \vdash T \leq T'$ and $\Gamma, t \leq T \vdash U \leq U'$
(VarTrans≤) $U \neq \text{Top}, U \neq t \Rightarrow$	$\Gamma \vdash t \leq U$	$\Leftrightarrow \Gamma \vdash \Gamma(t) \leq U$
(Top≤)		$\Gamma \vdash T \leq \text{Top}$
(Id≤)		$\Gamma \vdash t \leq t$
(Shape≤)	$\Gamma \vdash T \leq U$	$\Rightarrow \mathcal{S}(T) \leq \mathcal{S}(U)$
(μ≤l)	$\Gamma \vdash \mu X.T \leq U$	$\Leftrightarrow \Gamma \vdash [\mu X.T/X]T \leq U$
(μ≤r)	$\Gamma \vdash T \leq \mu X.U$	$\Leftrightarrow \Gamma \vdash T \leq [\mu X.U/X]U$

[7] We write $t \in \Gamma$ if $t \leq T$ is a component of Γ, for a certain T; similarly for $x \in \Gamma$.

(Top≤) and (Id≤) are immediate. The condition (Shape≤) is enforced by all the rules and is transmitted inductively by the (μ) rules. (VarTrans≤), in the ⇐ direction, is a consequence of the (Var≤) and (Trans≤) subtyping rules; the other ⇐ implications, including (Top≤) and (Id≤), are equivalent to the corresponding subtyping rules. Now we prove the ⇒ implications for (∀≤), (μ≤l/r) and (VarTrans≤).

(∀≤) This is the crucial case. To prove it we strengthen the condition to the conjunction of the following four conditions (where $L=\mu X.\forall t\leq T'.U$ and $R=\mu Y.\forall t\leq T.U'$):

(∀1) $\Gamma \vdash \forall t\leq T'.U \leq \forall t\leq T.U' \Rightarrow \Gamma \vdash T \leq T'$ and $\Gamma, t\leq T \vdash U \leq U'$

(∀2) $\Gamma \vdash \mu X.\forall t\leq T'.U \leq \forall t\leq T.U' \Rightarrow \Gamma \vdash T \leq [L/X]T'$ and $\Gamma, t\leq T \vdash [L/X]U \leq U'$

(∀3) $\Gamma \vdash \forall t\leq T'.U \leq \mu Y.\forall t\leq T.U' \Rightarrow \Gamma \vdash [R/Y]T \leq T'$ and $\Gamma, t\leq [R/Y]T \vdash U \leq [R/Y]U'$

(∀4) $\Gamma \vdash \mu X.\forall t\leq T'.U \leq \mu Y.\forall t\leq T.U'$
$\Rightarrow \Gamma \vdash [R/Y]T \leq [L/X]T'$ and $\Gamma, t\leq [R/Y]T \vdash [L/X]U \leq [R/Y]U'$

We denote the four conditions at once as follows, where (μX.)… means "optional recursion" and []U means "substitute the recursive variable in U, if needed" (this is just an informal notation to avoid repeating four times some proofs):

(∀*) $\Gamma \vdash (\mu X.)\forall t\leq T'.U \leq (\mu Y.)\forall t\leq T.U' \Rightarrow \Gamma \vdash []T \leq []T'$ and $\Gamma, t\leq []T \vdash []U \leq []U'$

We prove the four conditions by induction on the minimum depth of a proof proving

(*) $\Gamma \vdash_\mu (\mu X.)\forall t\leq T'.U \leq (\mu Y.)\forall t\leq T.U'$.

(*) can be proved by transitivity, by (∀) or by (unfold-l/r) (the (Shape≤) condition leaves no other possibility).

(Trans): (the shape of the intermediate type is determined by (Shape≤)):

$\Gamma \vdash_\mu (\mu X.)\forall t\leq T'.U \leq (\mu Z.)\forall t\leq R.S$ $\quad \Gamma \vdash_\mu (\mu Z.)\forall t\leq R.S \leq (\mu Y.)\forall t\leq T.U'$

⇒(by induction) $\quad \Gamma \vdash []R \leq []T' \quad \Gamma \vdash []T \leq []R$
$\Gamma, t\leq []R \vdash_\mu []U \leq []S \quad \Gamma, t\leq T \vdash_\mu []S \leq []U'$
⇒(by narrowing[8]) $\quad \Gamma, t\leq []T \vdash_\mu []U \leq []S$
⇒(by transitivity) $\quad \Gamma \vdash []T \leq []T', \quad \Gamma, t\leq []T \vdash_\mu []U \leq []U'$

(∀): The thesis is immediate

(unfold-r): $\dfrac{\Gamma \vdash_\mu \forall t\leq []T'.[]U \leq (\mu Y.)\forall t\leq T.U'}{\Gamma \vdash_\mu \mu X.\forall t\leq T'.U \leq (\mu Y.)\forall t\leq T.U'}$

By induction: $\Gamma \vdash []T \leq []T'$, $(\Gamma, t\leq []T) \vdash_\mu []U \leq []U'$; (unfold-r) is identical.

(μ≤l/r) $\Gamma \vdash \mu X.V \leq U \Rightarrow \Gamma \vdash [\mu X.V/X]V \leq U$

Let $\Gamma \vdash \mu X.V \leq U$. Either it has been proved by (unfold-l), and the thesis follows immediately, or it has been proved by transitivity from:

$\Gamma \vdash \mu X.V \leq T \quad \Gamma \vdash T \leq U$
⇒(induction) $\quad \Gamma \vdash [\mu X.V/X]V \leq T \quad \Gamma \vdash T \leq U$
⇒(Trans) $\quad \Gamma \vdash [\mu X.V/X]V \leq U$

[8] Narrowing: $\Delta \vdash T\leq U$ and $\Gamma \leq_\Gamma \Delta \Rightarrow \Gamma \vdash T\leq U$; proof hint: substitute any use of an axiom $\Delta,\Delta' \vdash t\leq\Delta(t)$ in the proof of $\Delta \vdash T\leq U$ with the transitivity composition of an axiom $\Gamma,\Delta' \vdash t\leq\Gamma(t)$ and a proof of $\Gamma,\Delta' \vdash \Gamma(t)\leq\Delta(t)$.

or it has been proved by (unfold-r) (U=μY.T), from:

$\Gamma \vdash \mu X.V \le [\mu Y.T/Y]T$
$\Rightarrow$(induction) $\Gamma \vdash [\mu X.V/X]V \le [\mu Y.T/Y]T$
$\Rightarrow$(unfold-r) $\Gamma \vdash [\mu X.V/X]V \le \mu Y.T (=U)$

(VarTrans$\le$) $U \neq Top, U \neq t, \ \Gamma \vdash t \le U \ \Rightarrow \ \Gamma \vdash \Gamma(t) \le U$

Let $\Gamma \vdash t \le U$. Either $\Gamma \vdash t \le U$ is proved by VarTrans (trivial) or by transitivity:
$\Gamma \vdash t \le V \quad \Gamma \vdash V \le U$;

If V=t, the thesis is immediate; otherwise, by (Shape), V≠Top:
$\Rightarrow$(induction) $\Gamma \vdash \Gamma(t) \le V \quad \Gamma \vdash V \le U$
$\Rightarrow$(Trans) $\Gamma \vdash \Gamma(t) \le U$

or $\Gamma \vdash t \le U$ is proved by (unfold-r) (U=μY.T), from:

$\Gamma \vdash t \le [\mu Y.T/Y]T$
$\Rightarrow$(induction) $\Gamma \vdash \Gamma(t) \le [\mu Y.T/Y]T$
$\Rightarrow$(unfold-r) $\Gamma \vdash \Gamma(t) \le \mu Y.T (=U)$

□.

Lemma 5.4: The $\sqsubseteq$ relation is co-R.E. but is not R.E.

Proof hint : The $\sqsubseteq$ relation is co-R.E. since its complement $\not\sqsubseteq$ is semidecidable, because it is semidecided by the finite failure of the standard type-checking algorithm. Since $\sqsubseteq$ is co-R.E, we prove that it is not R.E. by proving that it is not decidable. This can be proved using, essentially, the same proof used in [Pie] to show the undecidability of $\le$. We cannot rephrase here the whole Pierce's proof, but we will just give an outline.

Pierce defines a state machine, called the "rowing machine", whose halting problem is undecidable, since it is equivalent to the halting problem of two-counter Turing machines. Then he considers the following subset J^F of J (F stands for "Flattened")[9]:

$T^+ = Top \mid \forall x_1 \le T^-_1 \ldots \forall x_n \le T^-_n.\text{-}T^-$
$T^- = x \mid \forall x_1 \ldots \forall x_n.\text{-}T^+$
$J^F = \vdash T^- \le T^+$

he shows that the following two-rule reduction system is correct and complete for $F_{\le}$ subtyping restricted to J^F triples:

(FTop) $\vdash T^- \le Top \quad \triangleright$ **true**

(F∀-) $\vdash \forall x_1 \ldots \forall x_n.\text{-}T^+ \le \forall x_1 \le T^-_1 \ldots \forall x_n \le T^-_n.\text{-}T^- \quad \triangleright$
$\vdash [T^-_1/x_1 \ldots T^-_n/x_n]T^- \le [T^-_1/x_1 \ldots T^-_n/x_n]T^+$

finally he shows that every rowing machine can be encoded as a J^F triple which terminates if and only if the rowing machine does, which implies that termination of the two-rule system on J^F triples is undecidable, hence that the provability of J^F judgements is undecidable, and that subtyping on the whole $F_{\le}$ is undecidable.

Note that every J^F triple is either a provable or a diverging judgement; to prove undecidability of finite failure we just substitute J^F with a different subset J^G of J which contains only finitely failing or diverging triples, by susbstituting Top with a fresh variable y:

[9] The actual definition of J^F is slightly more complex: first, in every type ∀x1≤T1…xn≤Tn.-T, no xi can be free in any Ti+j; furthermore, all the quantifier prefixes ∀x1≤T1…xn≤Tn appearing in a unique judgement must have the same length.

$T^+ = y \mid \forall x_1 \leq T^-_1 \ldots \forall x_n \leq T^-_n . -T^-$
$T^- = x \text{ (with } x \neq y) \mid \forall x_1 \ldots \forall x_n . -T^+$
$J^G = y \leq Top \vdash T^- \leq T^+$

Now we can rephrase Pierce's proof by using the following reduction system, correct and complete on J^G with respect to finite failure:

(Gy) $y \leq Top \vdash T^- \not\leq y \quad \triangleright$ **true**

(G∀-) $y \leq Top \vdash \forall x_1 \ldots \forall x_n . -T^+ \not\leq \forall x_1 \leq T^-_1 \ldots \forall x_n \leq T^-_n . -T^- \quad \triangleright$
$y \leq Top \vdash [T^-_1/x_1 \ldots T^-_n/x_n]T^- \not\leq [T^-_1/x_1 \ldots T^-_n/x_n]T^+$

Now, by encoding rowing machines over J^G triples, we prove that finite failure of subtyping is undecidable on J^G, and hence is undecidable on the whole J.