Uniform PERs and Comportment Analysis

Alan Mycroft[1] and Kirsten Lackner Solberg[2]

[1] Computer Laboratory, Cambridge University,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, United Kingdom
Alan.Mycroft@cl.cam.ac.uk
[2] Dept. of Math. and Computer Science, Odense University, Denmark and
DAIMI, Aarhus University, Denmark
kls@daimi.aau.dk

Abstract. Hunt showed that the notion of PER-based strictness properties subsumed the incomparable notions of ideal- and projection-based properties on non-lifted value spaces. We extend Hunt's idea so it can encompass other *comportment* properties (such as totality) by separating the information and fixpoint orderings. We then define a class of *uniform* PERs (in the sense that they treat non-bottom ground elements, e.g. integers, identically) and then show how these can be hereditarily defined on function spaces. These (or subsets thereof) can be taken as a space of abstract values for abstract interpretation in the style of Nielson.

1 Introduction

Strictness properties are properties of functions between domains; in principle they are intended to capture the notion of how the function reacts to changes in the definedness of its arguments rather than changes between incomparable values of its arguments. A comparison to the notion of partial differentiation may prove fruitful. Since many properties conforming to this notion (e.g. totality) exclude $\perp$, we use the word *comportment property* as coined by Cousot and Cousot [4] to avoid abusing the epithet "strictness property" by allowing it to encompass totality. Here comportment is considered within a denotational formalism.

This work started as an experiment to discover the expressive power of a natural notion of uniform PER as a basis for describing comportment properties. It transpires that the framework of "higher order abstract interpretation" in [4] describes a similar set of comportment properties after being specialised as in the second part of that work. The interest is that PERs in the traditional abstract interpretation framework can express similar properties as simple properties in the higher-order framework. We express no preference between these routes; the aim was to understand better PER-based properties. Moreover we do not discuss implementation complexity.

For many years there were two forms of strictness property, the ideal-based form and the projection-based form [8, 2, 12, 6, 5, 4]. Cousot and Cousot [3] consider a Galois connection view of strictness properties.

Given domains D and E and a continuous function $f : D \to E$ (the denotational semantics of a program function) these properties can be summarised as follows.

Let I range over the *ideals* of D (non-empty Scott-closed sets) and J over those of E. The ideal-based strictness properties are those of the form

$$W^{\mathrm{ideal}}_{I,J}(f) \Leftrightarrow f(I) \subseteq J.$$

Similarly, let α range over projections on D (continuous, idempotent functions such that $\alpha(x) \sqsubseteq x$) and β over those on E. The projection-based strictness properties are those of the form

$$W^{\mathrm{proj}}_{\alpha,\beta}(f) \Leftrightarrow \beta \circ f = \beta \circ f \circ \alpha.$$

Hunt [6] observed that PER-based properties generalise both the above forms. (Actually Hunt only considered strict and inductive PER-properties which suffice for his generalisation but we relax this so as to be able to encompass as many comportment properties as possible.)

A PER P on D is a relation on D which is symmetric and transitive. It is inductive if it contains limits of chains when seen as a subset of $D \times D$. Such a PER P defines a property W^{PER}_P on D given by its diagonal

$$W^{\mathrm{PER}}_P(d) \Leftrightarrow d \in |P| \;\text{ where }\; |P| = \{d \mid dPd\}.$$

Hunt essentially defines the PER-properties of functions in two stages, first defining the basic PER-properties and then the PER-properties as the conjunctive completion (meet completion) of these. Letting P range over (a class of, see below) PERs of D and Q over (a class of) PERs of E, the basic PER-properties appear as

$$W^{\text{basic-PER}}_{P,Q}(f) \Leftrightarrow (\forall x, y \in D)(xPy \Rightarrow (fx)Q(fy)).$$

Conjunctions (intersections) of these give PER-properties.

The reader familiar with the presentations of abstract interpretation in the style of Nielson or Burn, Hankin and Abramsky *et al.* will here note an absence of specifying *how* the properties at higher-order are related to those at lower-order (or first-order). Indeed, Cousot and Cousot argue that the framework of abstract interpretation should leave this unspecified; only particular examples instantiating the framework should do this (see e.g. the two parts of [4]). In general the argument is that the framework should allow one to select representations for program properties at each type independently; all that is required is that function properties, ranged over by C, should be in Galois correspondence $\gamma : C \to \mathcal{P}(D \to E)$ where $\mathcal{P}(\cdot)$ is the power-*set* construct. This yields properties

$$W^{\mathrm{Galois}}_C(f) \Leftrightarrow f \in \gamma(C).$$

This paper leans towards the framework propounded by Nielson [9]. Here the framework requires that the properties on higher types are related to those on lower types (albeit possibly in different ways for different uses of a given type). This framework is tighter in that is forbids the parasitic possibility that, e.g., the space of properties for a type depends uncomputably on its type structure. On the other hand the attempt to forbid 'unreasonable' interpretations in the framework generally requires the framework to be repeatedly enhanced when desirable interpretations are found not to fit.

One can, perhaps over-simplistically, observe that the former view approach is platonic in that it prescribes a framework and then searches for instances which explain or yield various analyses (these instances appear always to have been decidable). Similarly the effect of the latter approach has generally been constructivist: abstract spaces have been crafted which explain exactly the range of phenomena at hand; abstract spaces for higher types are expected to be derivable from those at lower types (cf. intuitionistic implication). The old arguments transfer beautifully: the Platonist argues that the constructivist is doomed to extend his constructions each time the world demands (and never models it perfectly); the constructivist chides the generality of the Platonist for allowing parasitic solutions which have no application or physical reality.

The present construction of *uniform* PERs, both at ground type and hereditarily at higher types, is designed to capture as many comportment properties as possible (hopefully all) in a constructive manner. It was developed concurrently with [4] and attempts to capture comportments from the constructive, as opposed to platonic, viewpoint. Interestingly the two approaches differ (ours yields four extra properties at type $\mathtt{Int} \to \mathtt{Int}$). These are accounted for by the fact that uniform PERs naturally separate the notion of a function being total from it being constant; the application in [4] does not. Certainly, it is possible to imagine a variant of each which models the other.

The range of comportment properties expressible by uniform PERs include not only strictness properties but also totality properties. These latter are also captured by the annotated type system of Solberg *et al.* [10]. Note also the independent work by Benton [1] and Jensen [7] on expressing strictness properties in logical form.

1.1 Overview

In Section 2 we recall the definition of PERs then define the subset and Egli-Milner orderings on PERs.

Section 3 defines the notion of uniform PER on the integers. These PERs are uniform in the sense that they treat all the integers identically (as in [5]). We observe that some of the uniform PERs on the integers are *not* strict. Far from being a problem these non-strict PERs describe the property of being a non-bottom value (or at higher types being a total function) in contrast to most work on ideal- or projection-based program analysis (subject to the unsurprising need to use the Egli-Milner ordering on PERs which we define). The empty PER also appears as a uniform PER but is not proscribed because of its possible use to represent some sort of dead code. Next we form following Hunt the PERs on the function space and again we observe that some of these are not strict.

In Section 4 we take a closer look at the uniform PERs on $\mathbf{Z}_\perp \to \mathbf{Z}_\perp$. It appears that we are able to express all the properties that are expressible with ideal-based [5, 8], projection-based [12], and rule-based [1, 10] program analyses. Some new properties also emerge, e.g. the property of being constant on the integers, the property of being non-bottom on the integers, the property of being constant and non-bottom on the integers, the property of being non-bottom on the integers *or* the bottom-function. These include the comportment properties of [4].

For each property there exists an optimisation that can be applied to the code implementing an expression with that property. So again we can see the advantage of also considering the non-strict PERs. Moreover for each property there exists a function which has that property as the best/most exact description of the expression.

The next step is to define the language in Section 5, its standard semantics, and a non-standard semantics for program analysis. We define the semantics and the program analysis as two different interpretations of the language (as in [6]). Following Hunt's proof of correctness of the analysis we show that the standard and non-standard semantics of terms satisfy a ternary logical relation which relates pairs of standard values with an abstract value.

The final step is to define the abstract-denotations of the constants, e.g. *if*, *fix*, $+$, such that their standard-denotation is related to the abstract-denotation. The problems here is mostly for *fix*. Not all the PERs we are dealing with are strict therefore we cannot just follow Hunt using a union operator to form the fixpoint. (Starting with the least PER in the subset ordering (the empty PER) and applying the functional to it would result in the empty PER and hence the fixpoint would be the empty PER.) Following ideas dating from at least [8] we start with the least strict PER and a form of the Egli-Milner ordering which we define on PERs.

2 Formalism

We start by defining the notion of a PER on a set and then consider the possible orderings on PERs when the set has order-structure:

2.1 PERs on domains

Recall that a PER on a set S is a relation on S that is *symmetric* and *transitive*. Both the domain and range of the PER is equal to the diagonal-part of P, defined by $|P| = \{x \mid (x, x) \in P\}$. For a given set A of PERs, the *properties* associated with A are the set of diagonals of members of A. PERs can be ordered in at least two different ways: the subset ordering and the Egli-Milner ordering.

Definition 1 *The subset ordering*
The PER P is less than or equal to the PER Q, written $P \leq Q$, if $P \subseteq Q$.

2.2 The Egli-Milner ordering

The subset ordering does not take into account the structure of the domain on which the PERs are built. The least PER in the subset ordering is the empty PER. The result of applying a functional to the empty PER is the empty PER, therefore the fixpoint of a functional is the empty PER. What we want to do is to start with the least, strict, PER. But starting there may not yield a chain under the subset ordering. The obvious choice is to use the Egli-Milner ordering.

Let D be a cpo. We define the Egli-Milner ordering on the space of PERs over D by treating the PERs as subsets of $D \times D$.

Definition 2 *The Egli-Milner ordering on PERs over D*
Let P and Q be PERs over D. We define $P \sqsubseteq_{\mathrm{EM}}^{\mathrm{PER}} Q$ if $P \sqsubseteq Q$ when considered as subsets of $D \times D$, i.e.

$$(\forall p \in P \exists q \in Q : p \sqsubseteq_{D \times D} q) \wedge (\forall q \in Q \exists p \in P : p \sqsubseteq_{D \times D} q).$$

□

We also define a PER P being strict and downwards closed by inheritance from $D \times D$:

Definition 3 *Strict and downwards closed PERs*
A PER P on D is *strict* if $(\bot_D, \bot_D) \in P$; it is *downwards closed* P if it is downwards closed when seen as a subset of $D \times D$, i.e.

$$((d_1, d_2) \sqsubseteq (d_3, d_4) \wedge (d_3, d_4) \in P) \Rightarrow (d_1, d_2) \in P$$

□

For strict PERs (i.e. Hunt's work) we would expect the subset ordering to coincide with the Egli-Milner ordering:

Lemma 4 *Egli-Milner ordering on strict and downwards closed PERs*
For all strict PERs P on D and for all downwards closed PER Q on D:

$$P \sqsubseteq_{\mathrm{EM}}^{\mathrm{PER}} Q \Leftrightarrow P \leq Q$$

Proof We assume P is a strict PER on D and that Q is a downwards closed PER on D.
First we assume $P \sqsubseteq_{\mathrm{EM}}^{\mathrm{PER}} Q$ and show $P \leq Q$. We assume $(x, y) \in P$ then from Definition 2 part one we get that there exits $(x', y') \in Q$ such that $(x, y) \leq (x', y')$. Since Q is downwards closed it can be the case that $x' = x$ and $y' = y$. Therefore we have that P is a subset of Q.
Next assume $P \leq Q$ and show $P \sqsubseteq_{\mathrm{EM}}^{\mathrm{PER}} Q$. Since P is a subset of Q then for all $(x, y) \in P$ we have $(x, y) \in Q$ such that $(x, y) \leq (x, y)$ which is the first part of Definition 2.
We have $(\bot_D, \bot_D) \in P$ since P is strict. Therefore for all $(x, y) \in Q$ we have $(\bot_D, \bot_D) \leq (x, y)$, which is the second part of Definition 2. ■

3 Uniform PERs on Types

Although the intuition is to define a class of uniform PERs associated with a *domain* it turns out to be more natural to define the classes of uniform PERs associated with (the standard interpretation of) a *type*. (For the purposes of abstract interpretation representations of these PERs can be used as abstract values and it is too restrictive to insist that (accidentally) isomorphic domains described by different types should have identical sets of abstract values). We will continue to refer to (e.g.) "the set of uniform PERs on $\mathbb{Z}_\perp$" when this reads better.

We start with a set Σ, of types, ranged over by σ and τ:

$$\sigma ::= \mathtt{Int} \mid \sigma \to \sigma \mid \sigma \times \sigma$$

For each type σ there is a standard domain D^{S}_σ:

$$D^{\mathrm{S}}_{\mathtt{Int}} = \mathbb{Z}_\perp \qquad D^{\mathrm{S}}_{\sigma\to\tau} = D^{\mathrm{S}}_\sigma \to D^{\mathrm{S}}_\tau \qquad D^{\mathrm{S}}_{\sigma\times\tau} = D^{\mathrm{S}}_\sigma \times D^{\mathrm{S}}_\tau$$

where $\to$ is continuous function space. We now define a PER on a type:

Definition 5
Let σ be a type, then P is a PER on σ if P is a PER on D^{S}_σ. □

For each type σ we now define a finite set of *uniform PERs*, $\mathcal{U}(\sigma)$, consisting of certain PERs on σ. A uniform PER on the integers is a PER on the standard domain of the integers which treats all the integers in the same way. The reason for only looking at uniform PERs is that in a comportment analysis we are typically interested in knowing whether an expression evaluates to an integer or undefined value—not about it being any particular integer. The uniform PERs on the integers are:

Definition 6
A PER P on `Int` is *uniform* if, whenever π is a permutation on $\mathbb{Z}_\perp$ $(= D^{\mathrm{S}}_{\mathtt{Int}})$ leaving $\perp$ unchanged, then $\forall x, y \in \mathbb{Z}_\perp : (x,y) \in P \Leftrightarrow (\pi x, \pi y) \in P$. □

The set of uniform PERs on the integers, which we call $\mathcal{U}(\mathtt{Int})$, thus contains the following 7 elements:

- 1A $= \emptyset$
- 2A $= \{(x,x) \mid x \in \mathbb{Z}\}$
- 3A $= \mathbb{Z} \times \mathbb{Z}$
- 1B $= \{(\perp,\perp)\}$
- 2B $= \{(x,x) \mid x \in \mathbb{Z}_\perp\}$
- 3B $= \mathbb{Z} \times \mathbb{Z} \cup \{(\perp,\perp)\}$
- 4 $= \mathbb{Z}_\perp \times \mathbb{Z}_\perp$

and they are related by the subset ordering and the Egli-Milner ordering as in Fig. 1. For the strict PERs {1B, 2B, 3B, 4} we observe that the Egli-Milner ordering coincides with the subset ordering.

Note that $\mathcal{U}(\mathtt{Int})$ is intersection and union closed: intersection closedness is desirable for abstract interpretation as it ensures each (standard) value has a best abstract approximation. Union closedness ensures that no more information than necessary is lost on a merge resulting from *if then else* .

We define the *uniform properties* $\mathcal{UP}(\sigma)$ associated with σ in the obvious way:

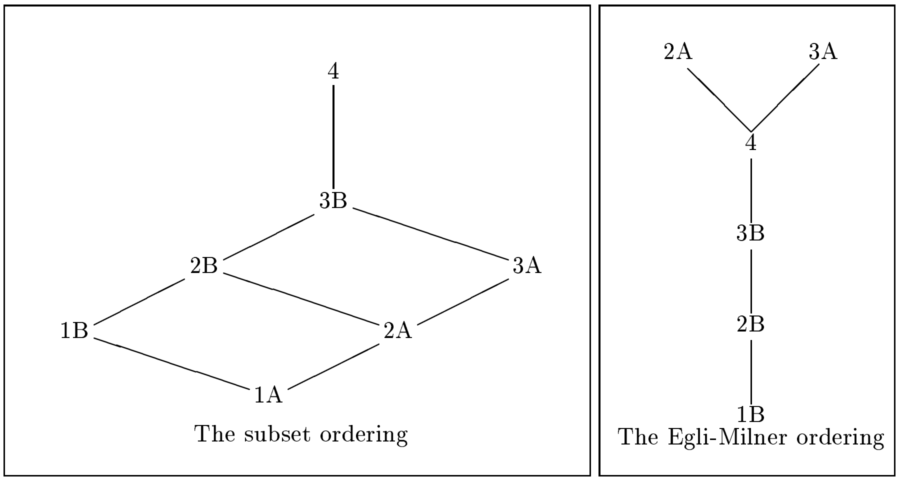

Fig. 1. The subset ordering and the Egli-Milner ordering on `Int`

Definition 7
To the set $\mathcal{U}(\sigma)$ of uniform PERs there is associated the set $\mathcal{UP}(\sigma)$ of properties:

$$\mathcal{UP}(\sigma) = \{|P| \mid P \in \mathcal{U}(\sigma)\}$$

$\square$

The uniform properties of `Int` are as one might expect: $\mathcal{UP}(\mathtt{Int}) = \{\emptyset, \{\perp\}, \mathbf{Z}, \mathbf{Z}_\perp\}$. These correspond exactly to the four values in [4], viz. $\emptyset$, $\perp$, $\not\perp$, $\top$. Note further that omitting PERs 2A and 2B from our definition of uniformity leads to the same property space on `Int`; moreover doing so seems to lead to the Cousots' comportment property space on $\mathtt{Int} \rightarrow \mathtt{Int}$.

Starting from $\mathcal{U}(\mathtt{Int})$ we will define uniform PERs on compound types. First we recall constructions which derive PERs at compound types from PERs of their components:

Definition 8
Given a PER P on σ and a PER Q on τ we can construct a PER $P\boxed{\rightarrow}Q$ on $\sigma \rightarrow \tau$ as:

$$P\boxed{\rightarrow}Q = \{(f, g) \in D^{\mathrm{S}}_{\sigma\rightarrow\tau} \times D^{\mathrm{S}}_{\sigma\rightarrow\tau} \mid (\forall (a, b) \in P)(fa, gb) \in Q\}$$

and a PER $P\boxed{\times}Q$ on $\sigma \times \tau$ as:

$$P\boxed{\times}Q = \{((a, c), (b, d)) \in D^{\mathrm{S}}_{\sigma\times\tau} \times D^{\mathrm{S}}_{\sigma\times\tau} \mid (a, b) \in P, (c, d) \in Q\}$$

(Note these are particular PERs, not spaces of PERs.) $\square$

Now we define the set of *uniform* PERs on compound types inductively. Doing so is facilitated by defining them mutually with a set of *basic* PERs.

Definition 9 *The uniform PERs on compound types*
Given types σ and τ and their associated sets $\mathcal{U}(\sigma)$ and $\mathcal{U}(\tau)$ of uniform PERs, we define, at type $\sigma \to \tau$, the set of basic PERs, $\mathcal{B}(\sigma \to \tau)$ as:

$$\mathcal{B}(\sigma \to \tau) = \{P \boxed{\to} Q \mid P \in \mathcal{U}(\sigma), Q \in \mathcal{U}(\tau)\}$$

and the set of uniform PERs, $\mathcal{U}(\sigma \to \tau)$ as:

$$\mathcal{U}(\sigma \to \tau) = \{\cap S \mid S \subseteq \mathcal{B}(\sigma \to \tau)\}.$$

Similarly we define, at type $\sigma \times \tau$, the set of basic PERs, $\mathcal{B}(\sigma \times \tau)$, as:

$$\mathcal{B}(\sigma \times \tau) = \{P \boxed{\times} Q \mid P \in \mathcal{U}(\sigma), Q \in \mathcal{U}(\tau)\}$$

and the set of uniform PERs, $\mathcal{U}(\sigma \times \tau)$, as:

$$\mathcal{U}(\sigma \times \tau) = \{\cup S \mid S \subseteq \mathcal{B}(\sigma \times \tau)\}.$$

$\square$

In general $\mathcal{B}(\sigma \to \tau)$ is not intersection closed; hence the definition of $\mathcal{U}(\sigma \to \tau)$ as its intersection-closure. Note that $\mathcal{U}(\sigma \to \tau)$ is not union closed. We resist the temptation to form the union-closure at higher types not from any polemic grounds but from a desire to preserve the "abstract function spaces are function spaces on abstract values" property found in [2, 5, 6].

Fact 10
Let σ be a type and A and B be members of $\mathcal{U}(\sigma)$ then: $A \cap B \in \mathcal{U}(\sigma)$ but in general $A \cup B \notin \mathcal{U}(\sigma)$. $\square$

Section 4 considers the uniform PERs and properties on `Int` $\to$ `Int` and the need for intersection closure in some detail.

For Observation 12 below we need the following lemma:

Lemma 11 *Intersections and unions*
For PERs P_1, P_2, ... P_n on σ and one PER Q on τ:

$$(P_1 \cup P_2 \cup \cdots \cup P_n) \boxed{\to} Q = (P_1 \boxed{\to} Q) \cap (P_2 \boxed{\to} Q) \cap \cdots \cap (P_n \boxed{\to} Q)$$

Proof Let

$$P_{\mathrm{L}} = (P_1 \cup P_2 \cup \cdots \cup P_n) \boxed{\to} Q$$

and

$$P_{\mathrm{R}} = (P_1 \boxed{\to} Q) \cap (P_2 \boxed{\to} Q) \cap \cdots \cap (P_n \boxed{\to} Q)$$

We show $P_{\mathrm{L}} = P_{\mathrm{R}}$ by the following chain of equivalences.

$$\begin{aligned}
(f, f') \in P_{\mathrm{R}} &\Leftrightarrow \forall i : [(d, d') \in P_i \Rightarrow ((fd, f'd') \in Q)] \\
&\Leftrightarrow (\forall i : (d, d') \in P_i) \Rightarrow ((fd, f'd') \in Q) \\
&\Leftrightarrow ((d, d') \in \cup P_i) \Rightarrow ((fd, f'd') \in Q) \\
&\Leftrightarrow (f, f') \in P_{\mathrm{L}}
\end{aligned}$$

Observation 12
We have

$$\begin{aligned}
&\mathcal{U}(\sigma_1 \times \sigma_2 \to \tau) \\
&= \{\cap S \mid S \subseteq \mathcal{B}(\sigma_1 \times \sigma_2 \to \tau)\} \\
&= \{\cap S \mid S \subseteq \{P \boxed{\to} Q \mid P \in \mathcal{U}(\sigma_1 \times \sigma_2), Q \in \mathcal{U}(\tau)\}\} \\
&= \{\cap S \mid S \subseteq \{P \boxed{\to} Q \mid P \in \{\cup S_1 \mid S_1 \subseteq \mathcal{B}(\sigma_1 \times \sigma_2)\}, Q \in \mathcal{U}(\tau)\}\}
\end{aligned}$$

Consider the uniform PERs on $\sigma_1 \times \sigma_2 \to \tau$ but with the restriction that the product PERs involved are basic PERs. That is look at:

$$\{\cap S \mid S \subseteq \{P \boxed{\to} Q \mid P \in \mathcal{B}(\sigma_1 \times \sigma_2), Q \in \mathcal{U}(\tau)\}\} = P_{\mathrm{B}}$$

Now since we have $\mathcal{B}(\sigma) \subseteq \mathcal{U}(\sigma)$ we arrive at $\mathcal{U}(\sigma_1 \times \sigma_2 \to \tau) \supseteq P_{\mathrm{B}}$. Showing $\mathcal{U}(\sigma_1 \times \sigma_2 \to \tau) \subseteq P_{\mathrm{B}}$ completes the proof, so consider a PER in $\mathcal{U}(\sigma_1 \times \sigma_2 \to \tau)$ in the form:

$$(P_{11} \cdots \cup P_{1n} \boxed{\to} Q_1) \cap \cdots \cap (P_{k1} \cup \cdots \cup P_{kn} \boxed{\to} Q_k)$$

where $P_{ij} \in \mathcal{B}(\sigma_1 \times \sigma_2)$ and $Q_i \in \mathcal{U}(\tau)$. By applying Lemma 11 we get

$$(P_{11} \boxed{\to} Q_1) \cap \cdots \cap (P_{1n} \boxed{\to} Q_1) \cap \cdots \cap (P_{k1} \boxed{\to} Q_k) \cap \cdots \cap (P_{kn} \boxed{\to} Q_k)$$

So we have that the PER is in P_{B} and therefore $\mathcal{U}(\sigma_1 \times \sigma_2 \to \tau) \subseteq P_{\mathrm{B}}$. $\square$

Following Observation 12, in this paper we restrict the types to the form

$$\sigma ::= \mathtt{Int} \mid \sigma \times \cdots \times \sigma \to \sigma$$

(This has the sole effect of forbidding products in function results and is done to avoid arguments concerning appropriate properties for product values.) Indeed, we could have achieved much the same effect by treating such restricted use of product types as shorthand for curried functions. However the present formalism is more natural and enables us to discuss properties of products. Note that because of Observation 12, it suffices to consider $\mathcal{B}(\sigma \times \tau)$ instead of $\mathcal{U}(\sigma \times \tau)$ since the restriction means that all products appear to the left of function types.

4 Examples in `Int` $\rightarrow$ `Int`

As an example let us take a look at the basic and uniform on `Int` $\rightarrow$ `Int`. PERs on `Int` $\rightarrow$ `Int` are subsets of the standard domain of (`Int` $\rightarrow$ `Int`) $\times$ (`Int` $\rightarrow$ `Int`). Given two uniform PERs P and Q on `Int` a *basic* PER on the function space `Int` $\rightarrow$ `Int` can be constructed as:

$$P\boxed{\rightarrow}Q = \{(f,g) \in (\mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp) \times (\mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp) \mid \forall (a,b) \in P \Rightarrow (fa, gb) \in Q\}$$

The set of all basic PERs on the function space `Int` $\rightarrow$ `Int` is:

$$\mathcal{B}(\texttt{Int} \rightarrow \texttt{Int}) = \{P\boxed{\rightarrow}Q \mid P, Q \in \mathcal{U}(\texttt{Int})\}$$

The set of all uniform PERs on the function space $\mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp$ is:

$$\mathcal{U}(\texttt{Int} \rightarrow \texttt{Int}) = \{\cap S \mid S \subseteq \mathcal{B}(\texttt{Int} \rightarrow \texttt{Int})\}$$

The reason why it is not sufficient only to use the basic PERs is that some expressions lack a least and best PER. Consider the term $\lambda x.\mathit{if}\ x = \perp\ \mathit{then}\ \perp\ \mathit{else}\ 42$; we have that

$$(\lambda x.\mathit{if}\ x = \perp\ \mathit{then}\ \perp\ \mathit{else}\ 42) \in |1\mathrm{B}\boxed{\rightarrow}1\mathrm{B}|$$

and

$$(\lambda x.\mathit{if}\ x = \perp\ \mathit{then}\ \perp\ \mathit{else}\ 42) \in |3\mathrm{A}\boxed{\rightarrow}2\mathrm{A}|.$$

However, the greatest lower bound in the subset ordering of PERs of the form $P\boxed{\rightarrow}Q$ of the two PERs 1B $\boxed{\rightarrow}$ 1B and 3A $\boxed{\rightarrow}$ 2A is the PER 3B $\boxed{\rightarrow}$ 1A which is empty. One of the missing PERs *is* the greatest lower bound in the subset ordering of 1B $\boxed{\rightarrow}$ 1B and 3A $\boxed{\rightarrow}$ 2A.

The properties, $\mathcal{UP}$(`Int` $\rightarrow$ `Int`), on `Int` $\rightarrow$ `Int` are:

- `empty` $= \emptyset$
- `strcon` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid (\forall a, b \in \mathbf{Z} : fa = fb \in \mathbf{Z}) \wedge (f\perp = \perp)\}$
- `ide` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid (\forall a \in \mathbf{Z} : fa \in \mathbf{Z}) \wedge (f\perp = \perp)\}$
- `div` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid \forall a \in \mathbf{Z}_\perp : fa = \perp\}$
- `strcon|div` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid (\forall a, b \in \mathbf{Z} : fa = fb \in \mathbf{Z}_\perp) \wedge (f\perp = \perp)\}$
- `ide|div` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid (\forall a \in \mathbf{Z} : fa \in \mathbf{Z}) \wedge f\perp = \perp\} \cup \texttt{div}$
- `strict` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid f\perp = \perp\}$
- `cgt` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid \forall a, b \in \mathbf{Z}_\perp : fa = fb \in \mathbf{Z}\}$
- `totcon` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid \forall a, b \in \mathbf{Z} : fa = fb \in \mathbf{Z}\}$
- `tot` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid \forall a \in \mathbf{Z} : fa \in \mathbf{Z}\}$
- `cgt|div` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid \forall a, b \in \mathbf{Z}_\perp : fa = fb \in \mathbf{Z}_\perp\}$
- `totcon|div` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid \forall a, b \in \mathbf{Z} : fa = fb \in \mathbf{Z}_\perp\}$
- `tot|div` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp \mid \forall a \in \mathbf{Z} : fa \in \mathbf{Z}\} \cup \texttt{div}$
- `all` $= \{f \in \mathbf{Z}_\perp \rightarrow \mathbf{Z}_\perp\}$

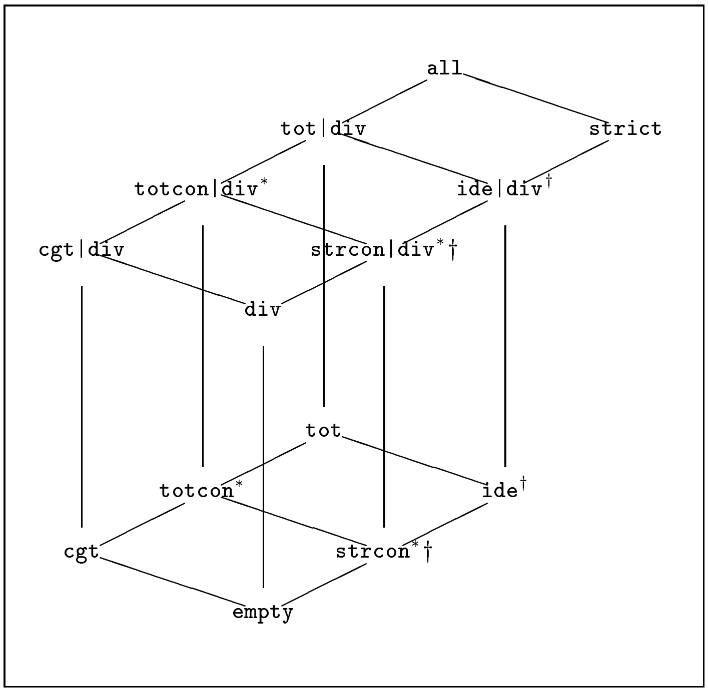

Fig. 2. The Subset ordering on $\mathcal{UP}(\texttt{Int} \to \texttt{Int})$

and they are related by the subset ordering as in Fig. 2 and by the Egli-Milner ordering as in Fig. 3.

In the figures, names have been chosen to match [4] except that '`cgt`' is used for 'convergent' leaving '`con`' to indicate 'constant'. Annotations: † indicates a PER added by conjunctive completion; * indicates a PER not in [4]. For PERs in [4] the '`|`' character in the name indicates it is added by disjunctive completion. Note that [4] has an error in that `ide|div` $\subseteq$ `tot|div` is omitted from the Hasse diagram.

We see that subset ordering coincides with the Egli-Milner ordering for the strict PERs (`div`, `cgt|div`, `strcon|div`, `totcon|div`, `ide|div`, `tot|div`, `strict`, and `all`). Also note that some of the PER-properties are Egli-Milner equivalent, viz.

$$\texttt{totcon|div} \equiv_{\mathrm{EM}} \texttt{cgt|div}$$
$$\texttt{strict} \equiv_{\mathrm{EM}} \texttt{ide|div}$$
$$\texttt{all} \equiv_{\mathrm{EM}} \texttt{tot|div}$$

This is also left unresolved in [4].

One question to be asked is: "are all these properties useful?". A property is useful if an optimisation is possible only when it holds (or, indirectly, if it

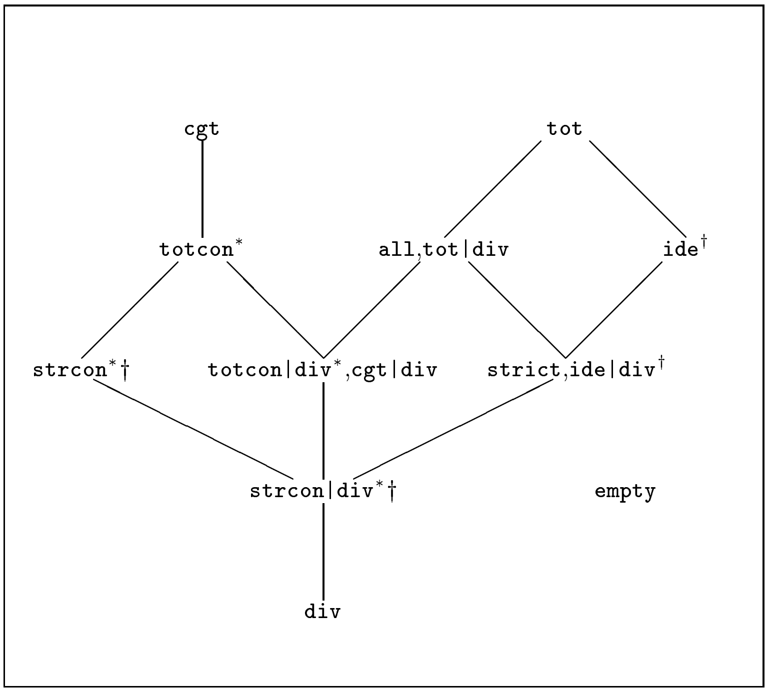

Fig. 3. The Egli-Milner ordering on $\mathcal{UP}(\texttt{Int} \to \texttt{Int})$

is needed to avoid information loss when calculating such a property—cf. joint strictness). Let f be an expression with property:

`empty` falsity property—which no function possesses.

`strcon` if the argument, a, is known to be an integer then $f0$ can be evaluated at compile-time and all these calls to f can be replaced by the result and if the argument is known not to terminate, then we can replace such calls with the result $\perp$.

`ide` if the argument is known not to terminate, then we can replace the calls with the result $\perp$ and in the case where we know that the argument, a, is an integer then fa can be evaluated at compile-time.

`div` replace all calls to f by the result $\perp$.

`strcon|div` if the argument is known not to terminate, then we can replace the calls with the result $\perp$ and if the argument, a, is known to be an integer then all such calls can be replaced by calls of e.g. $f0$.

`ide|div` Egli-Milner equivalent to its convex closure `strict` compared to which there do not seem to be additional optimisations.

`strict` evaluate argument before call.

`cgt`	evaluate $f\bot$ at compile-time and replace all calls to f with that result which is a terminating value.
`totcon`	if the argument, a, is known to be an integer then $f0$ can be evaluated at compile-time and all such calls to f can be replaced by the result.
`tot`	if the argument is known to be an integer then we can evaluate the call at compile-time.
`cgt|div`	replace all calls to f by $f\bot$; Egli-Milner equivalent to (and a subset of) `totcon|div`.
`totcon|div`	if the argument, a, is known to be an integer then calls fa can be replaced by calls of e.g. $f0$.
`tot|div`	there appear no useful optimisations for this property (which is perhaps unsurprising given that is it Egli-Milner equivalent to `all`).
`all`	truth property—which all functions possess.

The next question to ask is: "Do there exist terms which have each of these properties?". The term $\lambda x.4$ has the property `cgt` and it does not possess any property less than `cgt`. The property `cgt` is the best description of the term. Similarly $\lambda x.\bot$ has the property `div` which is the best. Temporarily suppose e' is the term *true*$\oplus$*false* where $\oplus$ denotes non-deterministic choice; then we can construct a term with the best property `cgt|div` viz. *if* e' *then* $\lambda x.4$ *else* $\lambda x.\bot$. This argument is more delicate in the absence of such an operator since any such e' must reduce to *true*, *false* or *bot*. However this fact is not discernible uniformly (for any analysis method there are undetectable tautologies) and hence for any analysis method there is such a term with best property `cgt|div`. The term *fix*$\lambda f.\lambda x.$*if x=0 then 1 else f(x-1)* has the property `strcon` as the best one. Now we are able to construct terms for the remainder as we did for the property `cgt|div`.

5 Implementation use

We assume a simple typed functional language, whose object types coincide with (and are modelled by) the meta-language types above. Its syntax is

$$e = x \mid c \mid ee' \mid \lambda x.e'$$

where x ranges over variables and c over constants. Its semantics is given in terms of type-indexed semantic functions $\mathcal{E}^I_\sigma$ by

$$\begin{aligned}
\mathcal{E}^I_\sigma[\![x]\!]\rho &= \rho x\\
\mathcal{E}^I_\sigma[\![c]\!]\rho &= c^I\\
\mathcal{E}^I_\tau[\![ee']\!]\rho &= \mathtt{app}^I_{\sigma\to\tau}(\mathcal{E}^I_{\sigma\to\tau}[\![e]\!]\rho)(\mathcal{E}^I_\sigma[\![e']\!]\rho)\\
\mathcal{E}^I_{\sigma\to\tau}[\![\lambda x.e']\!]\rho &= \mathtt{lam}^I_{\sigma\to\tau}(\lambda d.\mathcal{E}^I_\tau[\![e']\!]\rho[d/x])
\end{aligned}$$

where c^I (including $\mathtt{lam}^I_\sigma$ and $\mathtt{app}^I_\sigma$) are given by an interpretation which also specifies the interpretation of types as below.

For each type σ we have beside the standard domains D^{S}_σ an abstract domain D^{A}_σ. We will take D^{A}_σ to be the set of names of uniform PERs on σ; e.g. $D^{\mathrm{A}}_\sigma = \mathcal{N}(\sigma)$. The names of the uniform PERs are defined as follows:

$$\mathcal{N}(\mathtt{Int}) = \{1\mathrm{A}, 2\mathrm{A}, 3\mathrm{A}, 1\mathrm{B}, 2\mathrm{B}, 3\mathrm{B}, 4\}$$
$$\mathcal{N}(\sigma \to \tau) = \{[n_1, m_1; \ldots; n_k, m_k] \mid n_i \in \mathcal{N}(\sigma), m_i \in \mathcal{N}(\tau), k = |\mathcal{N}(\sigma)|\}$$
$$\text{in one name all the } n_i\text{'s are distinct}$$
$$\mathcal{N}(\sigma \times \tau) = \{(n, m) \mid n \in \mathcal{N}(\sigma), m \in \mathcal{N}(\tau)\}$$

Names on the function space $\sigma \to \tau$ can be seen as a graph of a function from names on σ to names on τ. For each type there is a concretisation function γ_σ from the set of names of PERs on the type, $\mathcal{N}(\sigma)$, to the set of uniform PERs on the type, $\mathcal{U}(\sigma)$. (Hunt refers to this type-indexed family of maps as *the logical concretisation map.*)

$$\gamma_\sigma \ : \ \mathcal{N}(\sigma) \to \mathcal{U}(\sigma)$$
$$\begin{array}{ll}
\gamma_{\mathtt{Int}}(1\mathrm{A}) = \emptyset & \gamma_{\mathtt{Int}}(1\mathrm{B}) = \{(\perp, \perp)\} \\
\gamma_{\mathtt{Int}}(2\mathrm{A}) = \{(x, x) \mid x \in \mathbf{Z}\} & \gamma_{\mathtt{Int}}(2\mathrm{B}) = \{(x, x) \mid x \in \mathbf{Z}_\perp\} \\
\gamma_{\mathtt{Int}}(3\mathrm{A}) = \mathbf{Z} \times \mathbf{Z} & \gamma_{\mathtt{Int}}(3\mathrm{B}) = \mathbf{Z} \times \mathbf{Z} \cup \{(\perp, \perp)\}
\end{array}$$
$$\gamma_{\mathtt{Int}}(4) = \mathbf{Z}_\perp \times \mathbf{Z}_\perp$$
$$\gamma_{\sigma\to\tau}([n_1, m_1; \ldots; n_k, m_k]) = (\gamma_\sigma n_1 \boxed{\to} \gamma_\tau m_1) \cap \ldots \cap (\gamma_\sigma n_k \boxed{\to} \gamma_\tau m_k)$$
$$\gamma_{\sigma\times\tau}((n, m)) = \gamma_\sigma n \boxed{\times} \gamma_\tau m$$

When using this representation (of functions by graphs) it is important to recall Hunt's observation that the γ are not injective—two (extensionally) different functions between properties may describe the same PER on the function space.

The Egli-Milner ordering on names of uniform PERs is inherited from the Egli-Milner ordering on the uniform PERs:

Definition 13
Let n and m be in $\mathcal{N}(\sigma)$ then $n \sqsubseteq^{\mathcal{N}}_{\mathrm{EM}} m$ if $\gamma_\sigma(n) \sqsubseteq^{\mathrm{PER}}_{\mathrm{EM}} \gamma_\sigma(m)$; $\sqcup^{\mathcal{N}}_{\mathrm{EM}}$ is similarly inherited (and well-defined as $\mathcal{N}(\sigma)$ is finite and closed over lubs). □

Standard and PER name-based interpretations are given in Fig. 4. The application and projection functions are monotonic and continuous:

Lemma 14
The two families of functions $\mathtt{app}^{\mathrm{S}}_{\sigma\to\tau}$ and $\mathtt{app}^{\mathrm{A}}_{\sigma\to\tau}$ are monotonic with respect to ordering on the standard domain, $\leq$, and the Egli-Milner ordering on names of PERs, $\sqsubseteq^{\mathcal{N}}_{\mathrm{EM}}$, respectively.

Proof From domain theory we know that $\mathtt{app}^{\mathrm{S}}_\sigma$ is monotonic. We prove that $\mathtt{app}^{\mathrm{A}}_\sigma$ is monotonic by induction on the type σ. For details see Solberg's PhD [11]. ■

Lemma 15
The functions π^{S}_1, π^{A}_1, π^{S}_2, and π^{A}_2 are monotonic with respect to ordering on the standard domain, $\leq$, and the Egli-Milner ordering on name of PERs, $\sqsubseteq^{\mathcal{N}}_{\mathrm{EM}}$, respectively.

$$
\begin{array}{ll}
\pi_1^{\mathrm{S}} : D^{\mathrm{S}}_{\sigma\times\tau} \to D^{\mathrm{S}}_{\sigma} & \pi_1^{\mathrm{A}} : D^{\mathrm{A}}_{\sigma\times\tau} \to D^{\mathrm{A}}_{\sigma} \\
\pi_1^{\mathrm{S}}((n,m)) = n & \pi_1^{\mathrm{A}}((n,m)) = n \\
\\
\pi_2^{\mathrm{S}} : D^{\mathrm{S}}_{\sigma\times\tau} \to D^{\mathrm{S}}_{\tau} & \pi_2^{\mathrm{A}} : D^{\mathrm{A}}_{\sigma\times\tau} \to D^{\mathrm{A}}_{\tau} \\
\pi_2^{\mathrm{S}}((n,m)) = m & \pi_2^{\mathrm{A}}((n,m)) = m \\
\\
\mathtt{app}^{\mathrm{S}}_{\sigma\to\tau} : D^{\mathrm{S}}_{\sigma\to\tau} \to D^{\mathrm{S}}_{\sigma} \to D^{\mathrm{S}}_{\tau} & \mathtt{app}^{\mathrm{A}}_{\sigma\to\tau} : D^{\mathrm{A}}_{\sigma\to\tau} \to D^{\mathrm{A}}_{\sigma} \to D^{\mathrm{A}}_{\tau} \\
\mathtt{app}^{\mathrm{S}}_{\sigma\to\tau}\ fx = fx & \mathtt{app}^{\mathrm{A}}_{\sigma\to\tau}\ [n_1, m_1; \ldots ; n_k, m_k]n_i = m_i \\
\\
\mathtt{lam}^{\mathrm{S}}_{\sigma\to\tau} : (D^{\mathrm{S}}_{\sigma} \to D^{\mathrm{S}}_{\tau}) \to D^{\mathrm{S}}_{\sigma\to\tau} & \mathtt{lam}^{\mathrm{A}}_{\sigma\to\tau} : (D^{\mathrm{A}}_{\sigma} \to D^{\mathrm{A}}_{\tau}) \to D^{\mathrm{A}}_{\sigma\to\tau} \\
\mathtt{lam}^{\mathrm{S}}_{\sigma\to\tau}\ f = f & \mathtt{lam}^{\mathrm{A}}_{\sigma\to\tau}\ f = [n_1, fn_1; \ldots ; n_k, fn_k], \\
 & \qquad \text{where } \ (n_i \in D^{\mathrm{A}}_{\sigma}) \wedge \\
 & \qquad (k = |D^{\mathrm{A}}_{\sigma}|) \\
\\
\mathtt{fix}^{\mathrm{S}}_{\sigma} : D^{\mathrm{S}}_{\sigma\to\sigma} \to D^{\mathrm{S}}_{\sigma} & \mathtt{fix}^{\mathrm{A}}_{\sigma} : D^{\mathrm{A}}_{\sigma\to\sigma} \to D^{\mathrm{A}}_{\sigma} \\
\mathtt{fix}^{\mathrm{S}}_{\sigma}\ f \quad = \sqcup \{d_i\} & \mathtt{fix}^{\mathrm{A}}_{\sigma}\ n \quad = \sqcup^{\mathcal{N}}_{\mathrm{EM}} \{d_i\} \\
\qquad d_0 = \perp_{D^{\mathrm{S}}_{\sigma}} & \qquad d_0 = \perp^{\mathrm{EM}\mathcal{N}}_{D^{\mathrm{A}}_{\sigma}} \\
\qquad d_{i+1} = \mathtt{app}^{\mathrm{S}}_{\sigma\to\sigma}\ fd_i & \qquad d_{i+1} = \mathtt{app}^{\mathrm{A}}_{\sigma\to\sigma}\ nd_i
\end{array}
$$

Fig. 4. Standard and non-standard interpretations for $\mathcal{E}^{\mathcal{I}}_{\sigma}$

Proof From domain theory we know that π_1^{S} and π_2^{S} are monotonic. We prove that π_1^{A} and π_2^{A} are monotonic by induction on the type $\sigma \times \tau$. For details see Solberg's PhD [11]. ∎

Lemma 16
The functions $\mathtt{app}^{\mathrm{S}}_{\sigma}$, $\mathtt{app}^{\mathrm{A}}_{\sigma}$, π_1^{S}, π_1^{A}, π_2^{S}, and π_2^{A} are continuous with respect to ordering on the standard domain, $\leq$, and the Egli-Milner ordering on name of PERs, $\sqsubseteq^{\mathcal{N}}_{\mathrm{EM}}$, respectively.

Proof It is a consequence of Lemma 14 and 15 and by recalling that monotonic functions from a space of finite height are continuous. ∎

5.1 Ternary logical relations

Now we can define the relations by:

$$(d, d') \in \gamma_\sigma(a) \Leftrightarrow (d, d')\mathcal{R}_\sigma a$$

Definition 17 *Ternary Logical Relations*
A type-indexed family of relations $(\mathcal{R}_\sigma)$ is logical (terminology conventionally abused by writing "$\mathcal{R}$ is a logical relation") if for all σ and τ:

$$
\begin{array}{l}
(f, f')\mathcal{R}_{\sigma\to\tau} h \Leftrightarrow (\forall d, d' \in D^{\mathrm{S}}_{\sigma}, \forall a \in D^{\mathrm{A}}_{\sigma}) \\
\qquad\qquad (d, d')\mathcal{R}_\sigma a \Leftrightarrow (\mathtt{app}^{\mathrm{S}}_{\sigma\to\tau}\ fd, \mathtt{app}^{\mathrm{S}}_{\sigma\to\tau}\ f'd')\mathcal{R}_\tau(\mathtt{app}^{\mathrm{A}}_{\sigma\to\tau}\ ha) \\
(p_1, p_2)\mathcal{R}_{\sigma\times\tau} p \Leftrightarrow (\pi_1^{\mathrm{S}}(p_1), \pi_1^{\mathrm{S}}(p_2))\mathcal{R}_\sigma \pi_1^{\mathrm{A}}(p) \wedge (\pi_2^{\mathrm{S}}(p_1), \pi_2^{\mathrm{S}}(p_2))\mathcal{R}_\tau \pi_2^{\mathrm{A}}(p)
\end{array}
$$

Proposition 18
The relation $\mathcal{R}$ is logical.

Proof We prove that $\mathcal{R}$ is a logical relation by induction on the type. For details see Solberg's PhD [11].

Proposition 19
The relation $\mathcal{R}$ is inductive with respect to ordering on the standard domain, $\leq$, and the Egli-Milner ordering on name of PERs, $\sqsubseteq^{\mathcal{N}}_{\mathrm{EM}}$, respectively.

Proof We prove that the relation $\mathcal{R}$ is inductive by induction on the type. For details see Solberg's PhD [11]. ∎

The standard fixpoint and the abstract fixpoint of related values are related:

Proposition 20
Let $(f, f)\mathcal{R}_{\sigma\to\sigma}h$ then $(\mathtt{fix}^{\mathrm{S}}_{\sigma} f, \mathtt{fix}^{\mathrm{S}}_{\sigma} f)\mathcal{R}_{\sigma}(\mathtt{fix}^{\mathrm{A}}_{\sigma} h)$

Proof We assume $(f, f)\mathcal{R}_{\sigma\to\sigma}h$. First we show that $(d^{\mathrm{S}}_i, d^{\mathrm{S}}_i)\mathcal{R}_{\sigma} d^{\mathrm{A}}_i$ holds for all i by induction on i. Next from Lemma 14 we know that $\mathtt{app}^{\mathrm{S}}$ and $\mathtt{app}^{\mathrm{A}}$ are monotonic therefore are $\{d^{\mathrm{S}}_i\}$ and $\{d^{\mathrm{A}}_i\}$ chains and since $\mathcal{R}$ is inductive (Proposition 19) we have $(\sqcup_i\{d^{\mathrm{S}}_i\}, \sqcup_i\{d^{\mathrm{S}}_i\})\mathcal{R}_{\sigma}(\sqcup^{\mathcal{N}}_{\mathrm{EM}}\{d^{\mathrm{A}}_i\})$ as required.

Case $i = 0$: We need to show $(\perp_{D^{\mathrm{S}}_{\sigma}}, \perp_{D^{\mathrm{S}}_{\sigma}})\mathcal{R}_{\sigma}\perp^{\mathrm{EM}\mathcal{N}}_{D^{\mathrm{A}}_{\sigma}}$ which is trivially implied by strictness of $\gamma_{\sigma}(\perp^{\mathrm{EM}\mathcal{N}}_{D^{\mathrm{A}}_{\sigma}})$.

Case $i + 1$: Assume $(d^{\mathrm{S}}_i, d^{\mathrm{S}}_i)\mathcal{R}_{\sigma} d^{\mathrm{A}}_i$ and show $(d^{\mathrm{S}}_{i+1}, d^{\mathrm{S}}_{i+1})\mathcal{R}_{\sigma} d^{\mathrm{A}}_{i+1}$: we have $d^{\mathrm{S}}_{i+1} = \mathtt{app}^{\mathrm{S}}_{\sigma}\ f d^{\mathrm{S}}_i$ and $d^{\mathrm{A}}_{i+1} = \mathtt{app}^{\mathrm{A}}_{\sigma}\ h d^{\mathrm{A}}_i$. Now, since $\mathcal{R}$ is logical (Proposition 18), we have $(\mathtt{app}^{\mathrm{S}}_{\sigma}\ f d^{\mathrm{S}}_i, \mathtt{app}^{\mathrm{S}}_{\sigma}\ f d^{\mathrm{S}}_i)\mathcal{R}_{\sigma}(\mathtt{app}^{\mathrm{A}}_{\sigma}\ h d^{\mathrm{A}}_i)$ as required.

Now the soundness and completeness of the analysis is:

Theorem 21
Let ρ be a standard environment and let δ be an abstract environment. Suppose for all constants c of type τ we have $(c^{\mathrm{S}}, c^{\mathrm{S}})\mathcal{R}_{\tau} c^{\mathrm{A}}$ then for all σ and e we have

$$(\rho, \rho)\mathcal{R}\delta \Rightarrow (\mathcal{E}^{\mathrm{S}}_{\sigma}[\![e]\!]\rho, \mathcal{E}^{\mathrm{S}}_{\sigma}[\![e]\!]\rho)\mathcal{R}_{\sigma}(\mathcal{E}^{\mathrm{S}}_{\sigma}[\![e]\!]\delta)$$

Proof We prove the Theorem by induction on the type. For details see Solberg's PhD [11]. ∎

Example 22
We calculate

$$\begin{aligned}
\mathcal{E}^{\mathrm{A}}_{\mathtt{Int}\to\mathtt{Int}}[\![\lambda x.4]\!]\rho &= \mathtt{lam}^{\mathrm{A}}_{\mathtt{Int}\to\mathtt{Int}}\ (\lambda d.\mathcal{E}^{\mathrm{A}}_{\mathtt{Int}}[\![\mathit{4}]\!]\rho[d/x]) \\
&= \mathtt{lam}^{\mathrm{A}}_{\mathtt{Int}\to\mathtt{Int}}\ (\lambda d.4^{\mathrm{A}}) \\
&= \mathtt{lam}^{\mathrm{A}}_{\mathtt{Int}\to\mathtt{Int}}\ (\lambda d.2\mathrm{A}) \\
&= [1\mathrm{A}, 2\mathrm{A}; 2\mathrm{A}, 2\mathrm{A}; 3\mathrm{A}, 2\mathrm{A}; 1\mathrm{B}, 2\mathrm{A}; 2\mathrm{B}, 2\mathrm{A}; 3\mathrm{B}, 2\mathrm{A}; 4, 2\mathrm{A}]
\end{aligned}$$

Now

$$\begin{aligned}\lambda x.4 &\in \mathcal{UP}(\gamma_{\texttt{Int}\to\texttt{Int}}[1\mathrm{A},2\mathrm{A};2\mathrm{A},2\mathrm{A};3\mathrm{A},2\mathrm{A};1\mathrm{B},2\mathrm{A};2\mathrm{B},2\mathrm{A};3\mathrm{B},2\mathrm{A};4,2\mathrm{A}]) \\ &= \texttt{cgt}\end{aligned}$$

as required. We also have

$$\begin{aligned}&\mathcal{E}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}[\![\mathit{fix}(\lambda x.4)]\!]\rho = \\ &\quad \texttt{fix}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}[1\mathrm{A},1\mathrm{B};2\mathrm{A},1\mathrm{B};3\mathrm{A},1\mathrm{B};1\mathrm{B},1\mathrm{B};2\mathrm{B},1\mathrm{B};3\mathrm{B},1\mathrm{B};4,1\mathrm{B}]\end{aligned}$$

and

$$\begin{aligned}d_0 &= 1\mathrm{B} \\ d_1 &= \texttt{app}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}\ [1\mathrm{A},2\mathrm{A};2\mathrm{A},2\mathrm{A};3\mathrm{A},2\mathrm{A};1\mathrm{B},2\mathrm{A};2\mathrm{B},2\mathrm{A};3\mathrm{B},2\mathrm{A};4,2\mathrm{A}]1\mathrm{B} \\ &= 2\mathrm{A} \\ d_2 &= \texttt{app}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}\ [1\mathrm{A},2\mathrm{A};2\mathrm{A},2\mathrm{A};3\mathrm{A},2\mathrm{A};1\mathrm{B},2\mathrm{A};2\mathrm{B},2\mathrm{A};3\mathrm{B},2\mathrm{A};4,2\mathrm{A}]2\mathrm{A} \\ &= 2\mathrm{A}\end{aligned}$$

and hence

$$\mathcal{E}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}[\![\mathit{fix}(\lambda x.4)]\!]\rho = 2\mathrm{A}$$

We calculate

$$\begin{aligned}\mathcal{E}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}[\![\lambda x.\bot]\!]\rho &= \texttt{lam}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}\ (\lambda d.\mathcal{E}^{\mathrm{A}}_{\texttt{Int}}[\![\bot]\!]\rho[d/x]) \\ &= \texttt{lam}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}\ (\lambda d.\bot^{\mathrm{A}}) \\ &= \texttt{lam}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}\ (\lambda d.1\mathrm{B}) \\ &= [1\mathrm{A},1\mathrm{B};2\mathrm{A},1\mathrm{B};3\mathrm{A},1\mathrm{B};1\mathrm{B},1\mathrm{B};2\mathrm{B},1\mathrm{B};3\mathrm{B},1\mathrm{B};4,1\mathrm{B}]\end{aligned}$$

and hence

$$\begin{aligned}\lambda x.\bot &\in \mathcal{UP}(\gamma_{\texttt{Int}\to\texttt{Int}}[1\mathrm{A},1\mathrm{B};2\mathrm{A},1\mathrm{B};3\mathrm{A},1\mathrm{B};1\mathrm{B},1\mathrm{B};2\mathrm{B},1\mathrm{B};3\mathrm{B},1\mathrm{B};4,1\mathrm{B}]) \\ &= \texttt{div}\end{aligned}$$

as required. We also have

$$\begin{aligned}&\mathcal{E}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}[\![\mathit{fix}(\lambda x.\bot)]\!]\rho = \\ &\quad \texttt{fix}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}[1\mathrm{A},1\mathrm{B};2\mathrm{A},1\mathrm{B};3\mathrm{A},1\mathrm{B};1\mathrm{B},1\mathrm{B};2\mathrm{B},1\mathrm{B};3\mathrm{B},1\mathrm{B};4,1\mathrm{B}]\end{aligned}$$

and

$$\begin{aligned}d_0 &= 1\mathrm{B} \\ d_1 &= \texttt{app}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}\ [1\mathrm{A},1\mathrm{B};2\mathrm{A},1\mathrm{B};3\mathrm{A},1\mathrm{B};1\mathrm{B},1\mathrm{B};2\mathrm{B},1\mathrm{B};3\mathrm{B},1\mathrm{B};4,1\mathrm{B}]1\mathrm{B} \\ &= 1\mathrm{B} \\ d_2 &= \texttt{app}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}\ [1\mathrm{A},1\mathrm{B};2\mathrm{A},1\mathrm{B};3\mathrm{A},1\mathrm{B};1\mathrm{B},1\mathrm{B};2\mathrm{B},1\mathrm{B};3\mathrm{B},1\mathrm{B};4,1\mathrm{B}]1\mathrm{B} \\ &= 1\mathrm{B}\end{aligned}$$

and hence $\mathcal{E}^{\mathrm{A}}_{\texttt{Int}\to\texttt{Int}}[\![\mathit{fix}(\lambda x.\bot)]\!]\rho = 1\mathrm{B}$.

6 Conclusion

We have defined the notion of uniform PERs on the integers and by following the framework of Hunt [6] we have lifted this notion to higher types in one particular way. In the work of Hunt all the PERs were strict. Here we have encountered PERs which are not strict; since they are useful too, we had to handle the fixpoint iteration in another way. We defined the Egli-Milner ordering on PERs and used it for the fixpoint iteration.

More work is need to clarify why our approach yields four extra properties at type $\texttt{Int} \to \texttt{Int}$ compared to the approach in [4]. The significance of some of the properties being Egli-Milner equivalent (both in this work and [4]) also needs more fuller investigation.

We now briefly compare this work with related papers. Firstly we show that the strictness properties of Burn, Hankin and Abramsky [2] naturally embed in the uniform PER properties presented here. To be precise, let $\mathcal{U}'(\texttt{Int}) = \{1\mathrm{B} = \{(\bot,\bot)\}, 4 = \mathbb{Z}_\bot \times \mathbb{Z}_\bot\}$ and $\mathcal{U}'(\sigma \to \tau)$ be defined inductively in the same manner as $\mathcal{U}(\sigma \to \tau)$. Then for all types σ we have $\mathcal{U}'(\sigma) \subseteq \mathcal{U}(\sigma)$ and moreover $\mathcal{U}'(\sigma)$ is isomorphic to $BHA(\sigma)$. Here $BHA(\sigma)$ is the set of strictness properties for type σ defined by Burn, Hankin and Abramsky:

$$\begin{aligned} BHA(\texttt{Int}) &= 2 = \{0,1\} \\ BHA(\sigma \to \tau) &= BHA(\sigma) \to BHA(\tau) \end{aligned}$$

Note that this embedding can also be seen as selecting uniform PER representatives of the uniform ideals of [5]. It also occurs as an abstraction of the comportment interpretation of [4] which abstracts our uniform PER interpretation.

We also compare with the properties based on annotated types described by Solberg *et al.* [10]. All the properties of that work can be modelled by the uniform PERs except for two: the property of knowing that a function does not have a WHNF and the property of knowing that a function does have a WHNF. This is not a lack of the uniform PER notion, but due to the fact that we have used non-lifted function spaces in which $\lambda x.\bot = \bot$ and hence WHNF properties are inexpressible.

Acknowledgements

Thanks to Radhia and Patrick Cousot for engaging arguments on the nature of abstract interpretation. Sebastian Hunt explained further about PERs in abstract interpretation. Thanks to Nick Benton, Hanne Riis Nielson, Flemming Nielson and Andrew Pitts for interesting discussions. We wish also to thank the PLILP referees for their comments which have aided the presentation. This work is supported by EU ESPRIT BRA 8130 "LOMAPS" and has been facilitated by Danish support for one author to visit Cambridge and by French support for the other to spend sabbatical leave at Ecole Polytechnique.

References

1. Nick Benton. *Strictness Analysis of Functional Programs.* PhD thesis, University of Cambridge, 1993. Available as Computer Laboratory Technical Report No. 309.
2. Geoffrey L. Burn, Chris Hankin and Samson Abramsky. Strictness analysis for higher-order functions. *Science of Computer Programming*, 7:249–278, 1986.
3. Patrick Cousot and Radhia Cousot. Galois connections based abstract interpretation for strictness analysis. In D. Bjørner, M. Broy and I. V. Pottosin, editors, *Proceedings of the International Conference on Formal Methods in Programming and their Applications*, LNCS 735, pages 98–127, 1993.
   [Currently available via `http://www.ens.fr/~cousot/`]
4. Patrick Cousot and Radhia Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), invited paper. In *Proceedings of the 1994 International Conference on Computer Languages, ICCL'94*, pages 95–112, IEEE Computer Society Press, 1994.
   [Currently available via `http://www.ens.fr/~cousot/`]
5. Christine Ernoult and Alan Mycroft. Uniform ideals and strictness analysis. In *ICALP 91, LNCS 510*, 1991.
   [Currently available via `http://www.cl.cam.ac.uk/users/am`]
6. Sebastian Hunt. *Abstract Interpretation of Functional Languages: From Theory to Practice.* PhD thesis, Imperial College, London, 1991.
7. Thomas P. Jensen. *Abstract Interpretation in Logical Form.* PhD thesis, Imperial College, London, 1992.
8. Alan Mycroft. *Abstract Interpretation and Optimising Transformation for Applicative programs.* PhD thesis, University of Edinburgh, 1981.
9. Flemming Nielson. Two-level semantics and abstract interpretation. *Theoretical Computer Science*, 69:117–242, 1989.
10. Kirsten Lackner Solberg, Hanne Riis Nielson and Flemming Nielson. Strictness and totality analysis. In *Proceedings of Static Analysis Symposium, LNCS 864*, 1994.
11. Kirsten Lackner Solberg. *Annotated Type Systems for Program Analysis.* PhD thesis, Odense University, to appear.
12. Philip Wadler and John Hughes. Projections for strictness analysis. In *Proceedings of Functional Programming Languages and Computer Architectures '87, LNCS 274*, 1987.