On Full Abstraction

Uwe Nestmann
Daniele Gorla

IFIP WG 2.2 @ Lisbon

Technische Universität Berlin
Modelle und Theorie Verteilter Systeme
Sapienza Università di Roma

Full Abstraction for Expressiveness: History, Myths and Facts†

Daniele Gorla[1] and Uwe Nestmann[2]

[1] *Dip. di Informatica, "Sapienza" Università di Roma.*
email: `gorla@di.uniroma1.it`

[2] *Technische Universitat Berlin.*
email: `uwe.nestmann@tu-berlin.de`

Received 20 December 2012

first attempt at a "panel discussion" during EXPRESS 2007 (CONCUR 2007) at Lisbon

History

Denotational Semantics

Denotational Semantics

"What is the meaning (semantics) of a program (syntax)?"

Denotational Semantics

"What is the meaning (semantics) of a program (syntax)?"

[[-]] : Syntax ➞ Math.Domain

Denotational Semantics

"What is the meaning (semantics) of a program (syntax)?"

$$[[\ -\ ]] : \text{Syntax} \rightarrow \text{Math.Domain}$$

full abstraction:

"*operational equivalence* coincides with *denotational equality*"

$$S_1 \simeq S_2 \quad \text{iff} \quad [[S_1]] = [[S_2]]$$

Denotational Semantics

"What is the meaning (semantics) of a program (syntax)?"

$[[\ -\ ]] : \text{Syntax} \rightarrow \text{Math.Domain}$

good for understanding

full abstraction:

"*operational equivalence* coincides with *denotational equality*"

$S_1 \simeq S_2 \quad \text{iff} \quad [[S_1]] = [[S_2]]$

Denotational Semantics

“What is the meaning (semantics) of a program (syntax)?”

$[\![-]\!] : \text{Syntax} \rightarrow \text{Math.Domain}$

good for understanding

full abstraction:

“*operational equivalence* coincides with *denotational equality*”

$$S_1 \simeq S_2 \quad \text{iff} \quad [\![S_1]\!] = [\![S_2]\!]$$

Milner 1997 in TCS 4 on models of Lambda Calculus
Plotkin 1977 in TCS 5 on PCF

Translational Semantics

Translational Semantics

"The meaning of a program (syntax) is another syntax."

[[-]] : Source.Syntax ➞ Target.Syntax

Translational Semantics

"The meaning of a program (syntax) is another syntax."

$[[-]] : \text{Source.Syntax} \rightarrow \text{Target.Syntax}$

"*operational equivalence* coincides
with *denotational equivalence*"

$S_1 \simeq S_2 \quad \text{iff} \quad [[S_1]] \simeq [[S_2]]$

Translational Semantics

"The meaning of a program (syntax) is another syntax."

[[-]] : Source.Syntax ➞ Target.Syntax

"*operational equivalence* coincides
with *denotational equivalence*"

$S_1 \simeq S_2$ iff $[[S_1]] \simeq [[S_2]]$

Riecke 1991 in POPL :
Fully abstract translations between functional languages

Translational Semantics

"The meaning of a program (syntax) is another syntax."

$[[\ -\ ]] :$ Source.Syntax ➞ Target.Syntax

good for proving

"*Operational equivalence* coincides with *denotational equivalence*"

$$S_1 \simeq S_2 \quad \text{iff} \quad [[S_1]] \simeq [[S_2]]$$

Riecke 1991 in POPL :
Fully abstract translations between functional languages

Translational Semantics

“The meaning of a program (syntax) is another syntax.”

$[[\ -\ ]] : \text{Source.Syntax} \rightarrow \text{Target.Syntax}$

good for proving

erational equivalence coincides
with *denotational equivalence*”

$S_1 \simeq S_2 \quad \text{iff} \quad [[S_1]] \simeq [[S_2]]$

Pandora’s
Box in C.T.

Riecke 1991 in POPL :
Fully abstract translations between functional languages

Myths

[Fournet, Gonthier 1996]

Definition 1 *Let $\mathcal{P}_1, \mathcal{P}_2$ be two process calculi, with respective equivalences $\approx_1 \subset \mathcal{P}_1 \times \mathcal{P}_1$, $\approx_2 \subset \mathcal{P}_2 \times \mathcal{P}_2$.*

$\mathcal{P}_2$ is more expressive *than $\mathcal{P}_1$ when there is a fully abstract encoding $[\![\ ]\!]_{1\to 2}$ from $\mathcal{P}_1$ to $\mathcal{P}_2$: for all P, Q in $\mathcal{P}_1$, we have*

$$P \approx_1 Q \iff [\![P]\!]_{1\to 2} \approx_2 [\![Q]\!]_{1\to 2}$$

$\mathcal{P}_1$ and $\mathcal{P}_2$ have the same expressive power *when each one is more expressive than the other.*

[Fournet, Gonthier 1996]

Definition 1 *Let $\mathcal{P}_1, \mathcal{P}_2$ be two process calculi, with respective equivalences $\approx_1 \subset \mathcal{P}_1 \times \mathcal{P}_1$, $\approx_2 \subset \mathcal{P}_2 \times \mathcal{P}_2$.*

$\mathcal{P}_2$ *is* more expressive *than* $\mathcal{P}_1$ *when there is a fully abstract encoding* $[\![\ ]\!]_{1 \to 2}$ *from* $\mathcal{P}_1$ *to* $\mathcal{P}_2$*: for all* P, Q *in* $\mathcal{P}_1$*, we have*

$$P \approx_1 Q \iff [\![P]\!]_{1 \to 2} \approx_2 [\![Q]\!]_{1 \to 2}$$

$\mathcal{P}_1$ *and* $\mathcal{P}_2$ *have the* same expressive power *when each one is more expressive than the other.*

We use observational congruence as the reference equivalence for each process calculus, meaning that our full abstraction results are up to observation in any context. This seems to be the finest results one could expect between different process calculi.

[Fournet, Gonthier 1996]

Definition 1 *Let $\mathcal{P}_1, \mathcal{P}_2$ be two process calculi, with respective equivalences $\approx_1 \subset \mathcal{P}_1 \times \mathcal{P}_1$, $\approx_2 \subset \mathcal{P}_2 \times \mathcal{P}_2$.*

$\mathcal{P}_2$ is more expressive *than $\mathcal{P}_1$ when there is a fully abstract encoding $[\![\ ]\!]_{1\rightarrow 2}$ from $\mathcal{P}_1$ to $\mathcal{P}_2$: for all P, Q in $\mathcal{P}_1$, we have*

$$P \approx_1 Q \iff [\![P]\!]_{1\rightarrow 2} \approx_2 [\![Q]\!]_{1\rightarrow 2}$$

$\mathcal{P}_1$ and $\mathcal{P}_2$ have the same expressive power *when each one is more expressive than the other.*

We use observational congruence as the reference equivalence for each process calculus, meaning that our full abstraction results are up to observation in any context. This seems to be the finest results one could expect between different process calculi.

Facts

Main Problem

the large choice of involved equivalences (especially in Concurrency Theory)

Main Problem

the large choice of involved equivalences (especially in Concurrency Theory)

much debate about *divergence-sensitiveness* !

Setting

An *encoding* $[\![\cdot]\!]$ of language $\mathbf{S} = (\mathcal{P}_{\mathbf{S}}, \longmapsto_{\mathbf{S}}, \simeq_{\mathbf{S}})$ into language $\mathbf{T} = (\mathcal{P}_{\mathbf{T}}, \longmapsto_{\mathbf{T}}, \simeq_{\mathbf{T}})$ is a (total) function $[\![\cdot]\!] : \mathcal{P}_{\mathbf{S}} \longrightarrow \mathcal{P}_{\mathbf{T}}$ mapping terms of $\mathcal{P}_{\mathbf{S}}$ into terms of $\mathcal{P}_{\mathbf{T}}$; by overloading, we also write $[\![\cdot]\!] : \mathbf{S} \longrightarrow \mathbf{T}$. We sometimes abbreviate $\mathcal{P}_{\mathbf{S}}$ and $\mathcal{P}_{\mathbf{T}}$ by $\mathcal{S}$ and $\mathcal{T}$. We let S and T range over terms of the source language ($\mathcal{S}$) and target language ($\mathcal{T}$), respectively.

Definition 1. *An encoding* $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *is* fully abstract *iff, for every* $S_1, S_2 \in \mathcal{P}_{\mathbf{S}}$*:*

$$(S_1 \simeq_{\mathbf{S}} S_2) \quad \Longleftrightarrow \quad ([\![S]\!]_1 \simeq_{\mathbf{T}} [\![S_2]\!])$$

$[\![\cdot]\!] : \mathcal{P}_{\mathbf{S}} \longrightarrow \mathcal{P}_{\mathbf{T}}$ *is then called* fully abstract w.r.t. $(\simeq_{\mathbf{S}}, \simeq_{\mathbf{T}})$.

False Positives

False Positives

Fact 1. *Let* $[\![\cdot]\!] : \mathcal{S} \to \mathcal{T}$ *be abitrary.*
Let $\mathbf{S} = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \mathrm{Ker}([\![\cdot]\!]))$ *for arbitrary* $\longmapsto_{\mathbf{S}}$.
Let $\mathbf{T} = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \mathrm{Id})$ *for arbitrary* $\longmapsto_{\mathbf{T}}$.
Then, $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *is fully abstract.*

False Positives

Fact 1. *Let* $[\![\cdot]\!] : \mathcal{S} \to \mathcal{T}$ *be abitrary.*
Let $\mathbf{S} = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \mathrm{Ker}([\![\cdot]\!]))$ *for arbitrary* $\longmapsto_{\mathbf{S}}$.
Let $\mathbf{T} = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \mathrm{Id})$ *for arbitrary* $\longmapsto_{\mathbf{T}}$.
Then, $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *is fully abstract.*

Fact 2. *Let* $[\![\cdot]\!] : \mathcal{S} \to \mathcal{T}$ *with* $[\![S]\!] = T$*, for all* $S \in \mathcal{S}$ *and some* $T \in \mathcal{T}$.
Let $\mathbf{S} = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \mathcal{S} \times \mathcal{S})$ *for arbitrary* $\longmapsto_{\mathbf{S}}$.
Let $\mathbf{T} = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq_{\mathbf{T}})$ *for arbitrary* $\longmapsto_{\mathbf{T}}$ *and* $\simeq_{\mathbf{T}}$.
Then, $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *is fully abstract.*

False Positives

Fact 3. *Let* TM *and* FA *denote the sets of Turing machines and of finite automata. Let* $\mathbf{TM} = (\mathrm{TM}, \longmapsto_{\mathrm{TM}}, \simeq_{\mathrm{TM}})$ *and* $\mathbf{FA} = (\mathrm{FA}, \longmapsto_{\mathrm{FA}}, \simeq_{\mathrm{FA}})$ *be defined with their standard operational semantics (viz.,* $\longmapsto_{\mathrm{TM}}$ *and* $\longmapsto_{\mathrm{FA}}$*) and language equivalence (viz.,* $\simeq_{\mathrm{TM}}$ *and* $\simeq_{\mathrm{FA}}$*). Then, there exists a fully abstract encoding* $[\![\cdot]\!] : \mathbf{TM} \to \mathbf{FA}$*.*

True Positives?

$$\llbracket\, \overline{a}\langle b, c\rangle.P \,\rrbracket \quad = \quad (\nu\, d)\overline{a}\langle d\rangle.\overline{d}\langle b\rangle.\overline{d}\langle c\rangle.\llbracket\, P \,\rrbracket$$

$$\llbracket\, a(x, y).Q \,\rrbracket \quad = \quad a(z).z(x).z(y).\llbracket\, Q \,\rrbracket$$

perfectly fine encoding, but not fully abstract

can be made fully abstract
by cheating on
the considered target contexts

True Negatives?

$$\mathbf{L}_1 = (\mathrm{CCS}, \longmapsto, \approx)$$
$$\mathbf{L}_2 = (\mathrm{CCS}, \longmapsto, \approx^\circ)$$
$$\mathbf{L}_3 = (\mathrm{CCS}_{gc}, \longmapsto, \approx)$$

Fact 4. *The embedding encoding of* $\mathbf{L}_3$ *into* $\mathbf{L}_2$ *does not preserve equivalences.*

Fact 5. *The identity encoding of* $\mathbf{L}_2$ *into* $\mathbf{L}_1$ *does not reflect equivalences.*

True Negatives?

Fact 6. *Consider the encoding of the asynchronous* π*-calculus with* $\approx_a$ *into the synchronous* π*-calculus with* $\approx$ *such that*

$$[\![\, \overline{a}\langle b\rangle \,]\!] = \overline{a}\langle b\rangle.\mathbf{0}$$

and homomorphic on all the other operators. Such an encoding does not preserve equivalences.

Proof. Consider $P = \mathbf{0}$ and $Q = a(x).\overline{a}\langle x\rangle$; it is well-known (Amadio et al. 1998) that $P \approx_a Q$, but $[\![\, P \,]\!] \not\approx [\![\, Q \,]\!]$. □

Changing Equivalences

Source Only

First of all, for a fully abstract encoding, one cannot change only the *source* equivalence without breaking full abstraction, be it by weakening or strenghtening of the equivalence.

Fact 7. *Let* $\mathbf{S} = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq_{\mathbf{S}})$, $\mathbf{T} = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq_{\mathbf{T}})$ *and* $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *fully abstract.*

1 *Let* $\simeq'_{\mathbf{S}} \subset \simeq_{\mathbf{S}}$ *and* $\mathbf{S}' = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq'_{\mathbf{S}})$. *Then, the encoding* $[\![\cdot]\!] : \mathbf{S}' \to \mathbf{T}$ *is not fully abstract.*

2 *Let* $\simeq'_{\mathbf{S}} \supset \simeq_{\mathbf{S}}$ *and* $\mathbf{S}' = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq'_{\mathbf{S}})$. *Then, the encoding* $[\![\cdot]\!] : \mathbf{S}' \to \mathbf{T}$ *is not fully abstract.*

Changing Equivalences

Target Only

By contrast, it is possible to change only the target equivalence without breaking full abstraction only if the encoding is not surjective (as it is usually the case). For surjective encodings, a situation similar to Fact 7 holds.

Fact 8. *Let* $\mathbf{S} = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq_{\mathbf{S}})$, $\mathbf{T} = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq_{\mathbf{T}})$ *and* $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *fully abstract and not surjective. Then, there exists a* $\simeq'_{\mathbf{T}}$ *different from* $\simeq_{\mathbf{T}}$ *such that* $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}'$, *for* $\mathbf{T}' = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq'_{\mathbf{T}})$, *is still fully abstract.*

Changing Equivalences

Both Source and Target

Fact 9. *Let* $\mathbf{S} = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq_{\mathbf{S}})$, $\mathbf{T} = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq_{\mathbf{T}})$ *and* $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *fully abstract and injective. Then, for every* $\simeq'_{\mathbf{S}} \subset \simeq_{\mathbf{S}}$, *there exists* $\simeq'_{\mathbf{T}} \subset \simeq_{\mathbf{T}}$ *such that the encoding* $[\![\cdot]\!] : \mathbf{S}' \to \mathbf{T}'$ *is fully abstract, where* $\mathbf{S}' = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq'_{\mathbf{S}})$ *and* $\mathbf{T}' = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq'_{\mathbf{T}})$.

Fact 10. *Let* $\mathbf{S} = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq_{\mathbf{S}})$, $\mathbf{T} = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq_{\mathbf{T}})$ *and* $[\![\cdot]\!] : \mathbf{S} \to \mathbf{T}$ *fully abstract. Then, for every* $\simeq'_{\mathbf{S}} \supset \simeq_{\mathbf{S}}$, *there exists* $\simeq'_{\mathbf{T}} \supset \simeq_{\mathbf{T}}$ *such that the encoding* $[\![\cdot]\!] : \mathbf{S}' \to \mathbf{T}'$ *is fully abstract, where* $\mathbf{S}' = (\mathcal{S}, \longmapsto_{\mathbf{S}}, \simeq'_{\mathbf{S}})$ *and* $\mathbf{T}' = (\mathcal{T}, \longmapsto_{\mathbf{T}}, \simeq'_{\mathbf{T}})$.

Conclusions ?

Pros & Cons

full abstraction

- may well be informative
  to discuss "aspects" of expressive power
- is (alone) useless for separation results ...