On Compositional Compiler Correctness and Fully Abstract Compilation

Daniel Patterson and Amal Ahmed

January 13, 2018

Northeastern University

What is compiler correctness?

What is compiler correctness?

If s compiles to t,

then s has the same behavior as t.

$$\boxed{s \leadsto t \implies s \approx t}$$

What is compiler correctness?

If s compiles to t,

then s has the same behavior as t.

$$\boxed{s \rightsquigarrow t \implies s \approx t}$$

How is this expressed?

How is $\boxed{s \approx t}$ expressed?

For whole-program compilers:

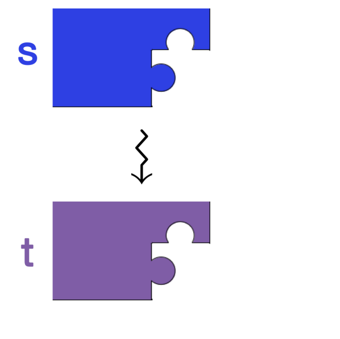
if running s produces behavior $\mathcal{O}$,

then running t should also produce $\mathcal{O}$.

$$\boxed{s \Downarrow \mathcal{O} \implies t \Downarrow \mathcal{O}}$$

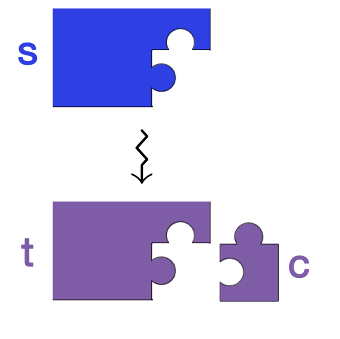
How is $\boxed{s \approx t}$ expressed?

Problem: components can't be run.



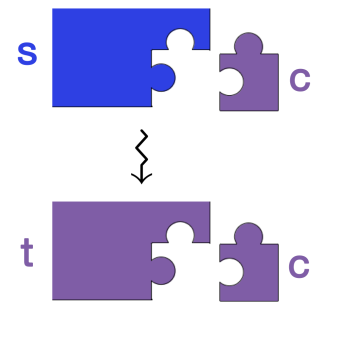
How is $\boxed{s \approx t}$ expressed?

Problem: components can't be run.



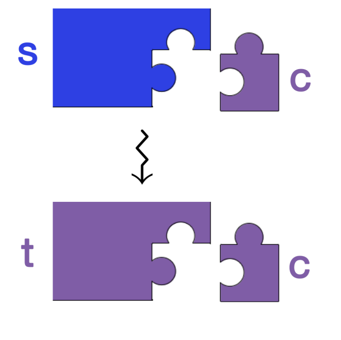
How is $\boxed{s \approx t}$ expressed?

Problem: components can't be run.



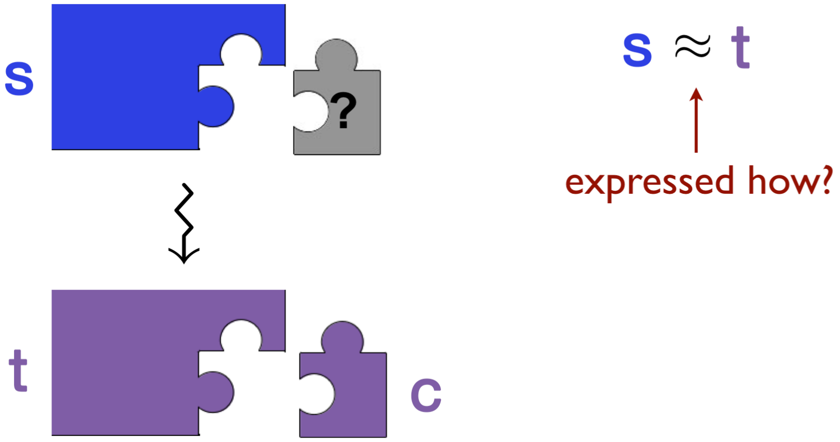
How is $\boxed{s \approx t}$ expressed?

Problem: components can't be run.

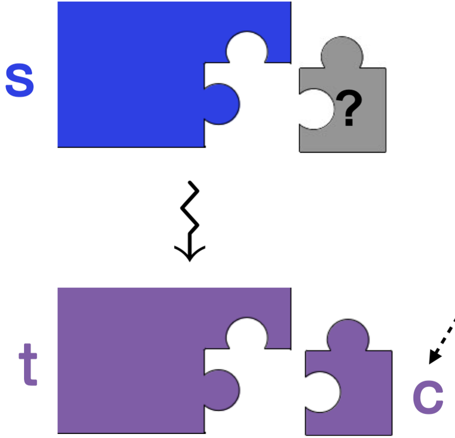


$$\boxed{c \ltimes s \Downarrow \mathcal{O} \implies c \ltimes t \Downarrow \mathcal{O}}$$

How is $\boxed{s \approx t}$ expressed?



How is $\boxed{s \approx t}$ expressed?

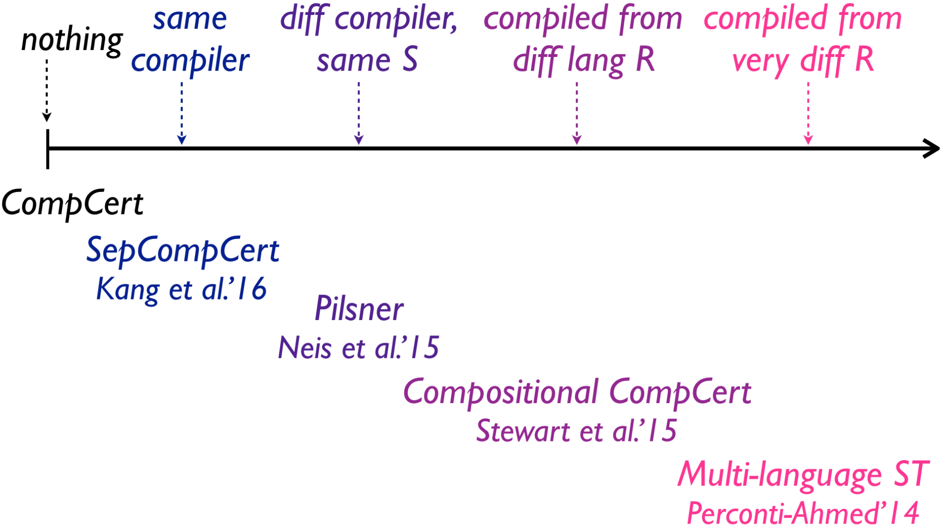


$s \approx t$

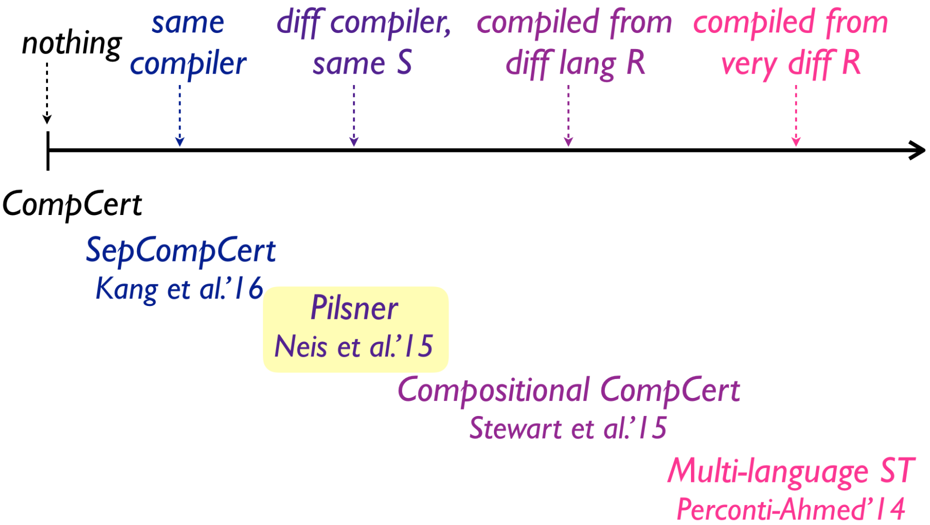
expressed how?

Produced by
- *same compiler,*
- *diff compiler for S,*
- *compiler for diff lang R,*
- *R that's **very** diff from S?*

Survey of some recent results

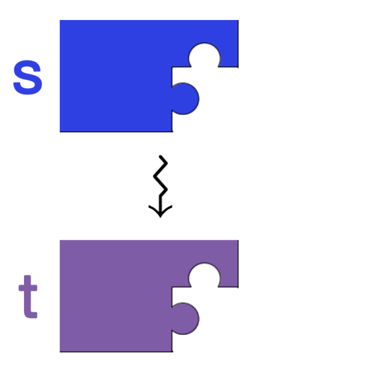


Survey of some recent results



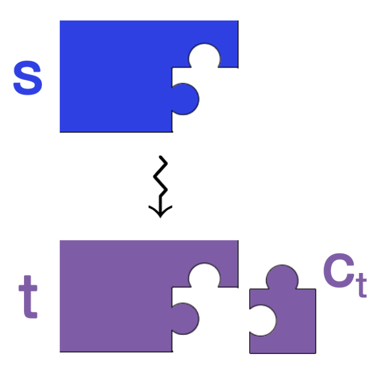
Approach: Pilsner

Neis, Hur, Kaiser, McLaughlin, Dreyer, Vafeiadis. ICFP 2015. *Pilsner: A Compositionally Verified Compiler for a Higher-Order Imperative Language.*



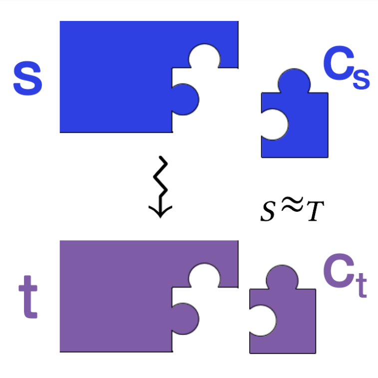
Approach: Pilsner

Neis, Hur, Kaiser, McLaughlin, Dreyer, Vafeiadis. ICFP 2015. *Pilsner: A Compositionally Verified Compiler for a Higher-Order Imperative Language.*



Approach: Pilsner

Neis, Hur, Kaiser, McLaughlin, Dreyer, Vafeiadis. ICFP 2015. *Pilsner: A Compositionally Verified Compiler for a Higher-Order Imperative Language.*



Correctness Theorem:

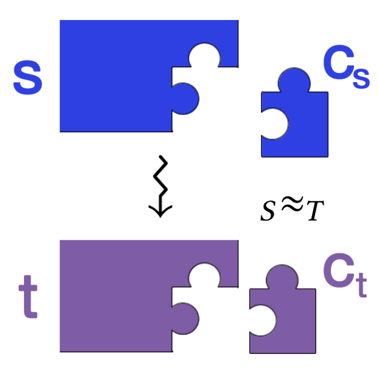
$\forall s \rightsquigarrow t. \forall c_S \; S{\approx}_T \; c_T.$

$c_S \ltimes s \Downarrow \mathcal{O} \implies$

$c_T \ltimes t \Downarrow \mathcal{O}.$

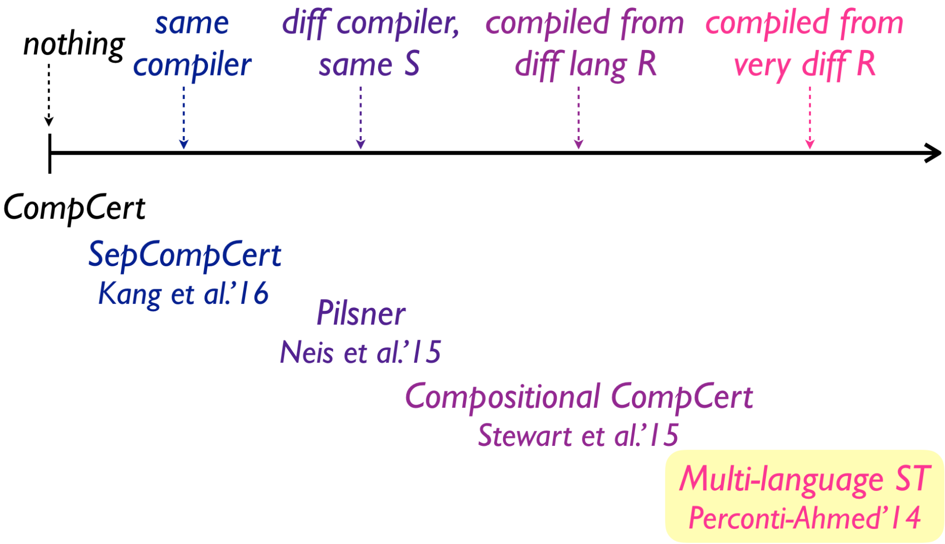
Approach: Pilsner

Neis, Hur, Kaiser, McLaughlin, Dreyer, Vafeiadis. ICFP 2015. *Pilsner: A Compositionally Verified Compiler for a Higher-Order Imperative Language.*



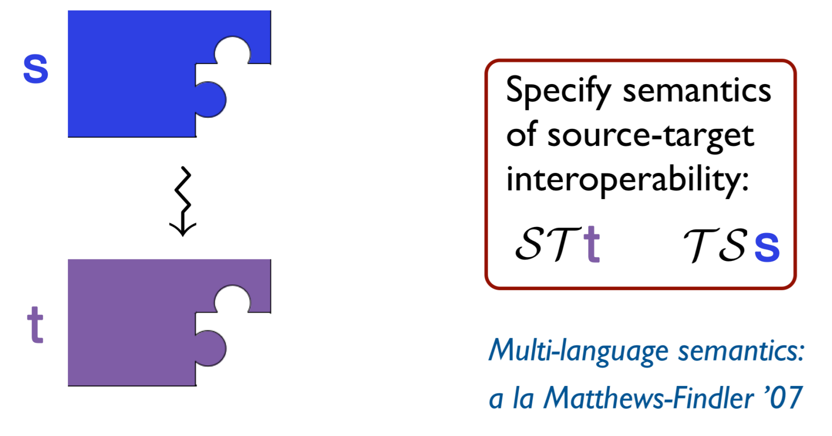
To link with target language c_T, need to produce related source language c_S.

Survey of some recent results



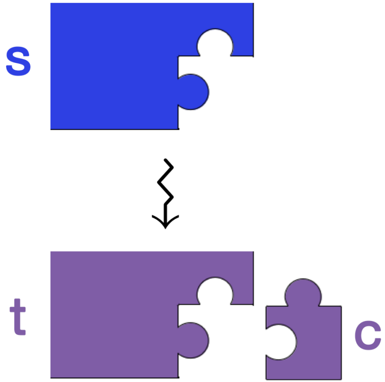
Approach: Multi-language

Perconti, Ahmed. ESOP 2014. *Fully Abstract Compilation via Universal Embedding.*



Approach: Multi-language

Perconti, Ahmed. ESOP 2014. *Fully Abstract Compilation via Universal Embedding.*



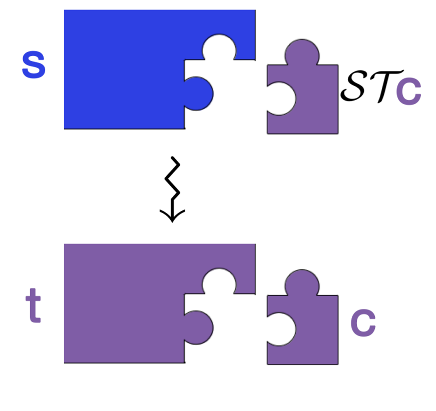
Specify semantics of source-target interoperability:

$\mathcal{ST}\,\mathbf{t}$ $\qquad$ $\mathcal{TS}\,\mathbf{s}$

Multi-language semantics: a la Matthews-Findler '07

Approach: Multi-language

Perconti, Ahmed. ESOP 2014. *Fully Abstract Compilation via Universal Embedding.*



Correctness Theorem:

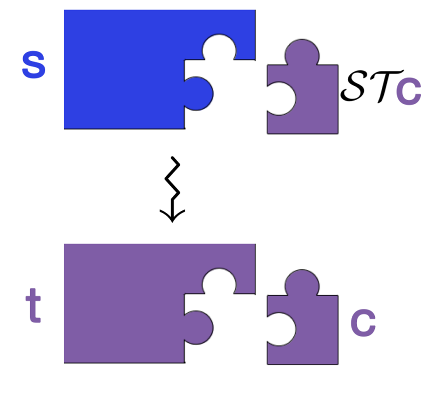
$\forall \mathsf{s} \rightsquigarrow \mathsf{t}. \forall \mathsf{c}.$

$\mathcal{ST}\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies$

$\mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}.$

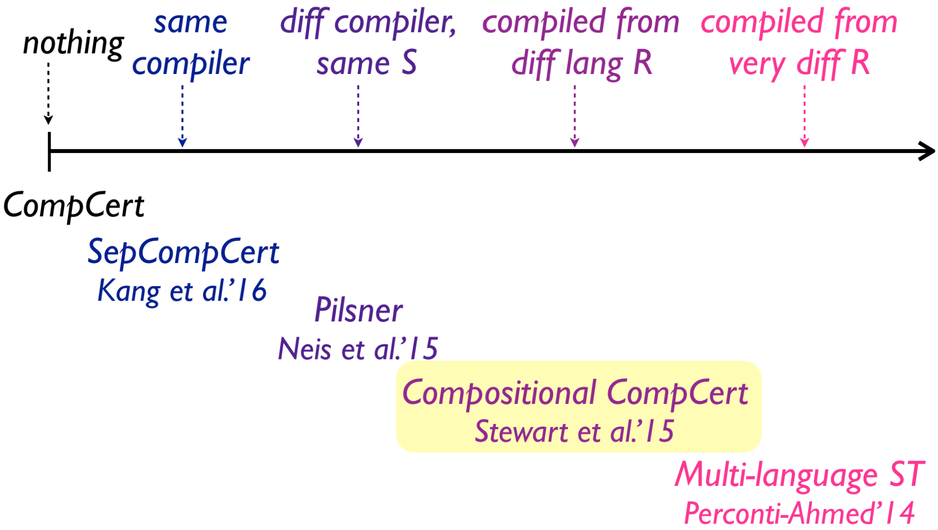
Approach: Multi-language

Perconti, Ahmed. ESOP 2014. *Fully Abstract Compilation via Universal Embedding.*



To add compiler passes, new multi-language must be created & formalized.

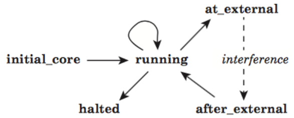
Survey of some recent results



Approach: CompComp

Stewart, Beringer, Cuellar, Appel. POPL 2015.
Compositional CompCert.

Language-independent linking of C-like langs.

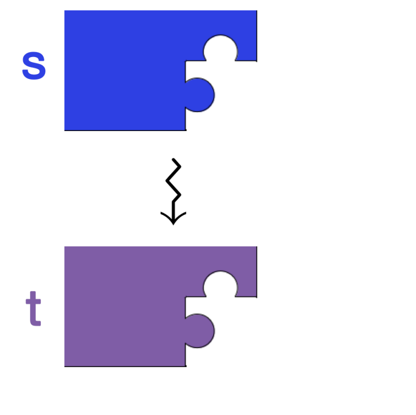


$$
\mathsf{Semantics}\ (G\ C\ M : \mathsf{Type})\ :\ \mathsf{Type} \triangleq
\left\{
\begin{array}{lcl}
\mathsf{initial\_core} & : & G \to \mathcal{V} \to \mathsf{list}\ \mathcal{V} \to \mathsf{option}\ C \\
\mathsf{at\_external} & : & C \to \mathsf{option}\ (\mathcal{F} \times \mathsf{list}\ \mathcal{V}) \\
\mathsf{after\_external} & : & \mathsf{option}\ \mathcal{V} \to C \to \mathsf{option}\ C \\
\mathsf{halted} & : & C \to \mathsf{option}\ \mathcal{V} \\
\mathsf{corestep} & : & G \to C \to M \to C \to M \to \mathsf{Prop}
\end{array}
\right.
$$

Figure 2. Interaction semantics interface. The types G (global environment), C (core state), and M (memory) are parameters to the interface. $\mathcal{F}$ is the type of external function identifiers. $\mathcal{V}$ is the type of CompCert values.

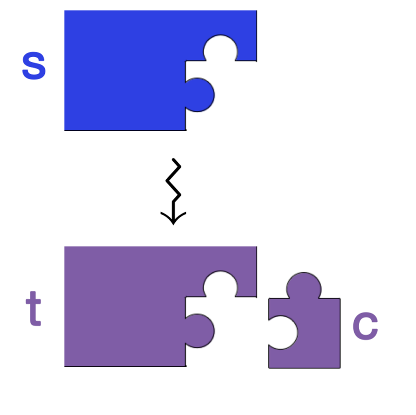
Approach: CompComp

Stewart, Beringer, Cuellar, Appel. POPL 2015.
Compositional CompCert.



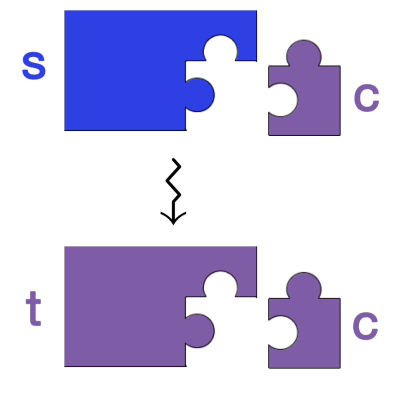
Approach: CompComp

Stewart, Beringer, Cuellar, Appel. POPL 2015.
Compositional CompCert.



Approach: CompComp

Stewart, Beringer, Cuellar, Appel. POPL 2015.
Compositional CompCert.



Correctness Theorem:

$\forall s \rightsquigarrow t. \forall c.$

$c \ltimes s \Downarrow \mathcal{O} \implies$

$c \ltimes t \Downarrow \mathcal{O}.$

Problem this research addresses

To understand if Theorem is correct...

Pilsner	source-target PILS relation
CompComp	interaction semantics
Multi-lang	source-target multi-language

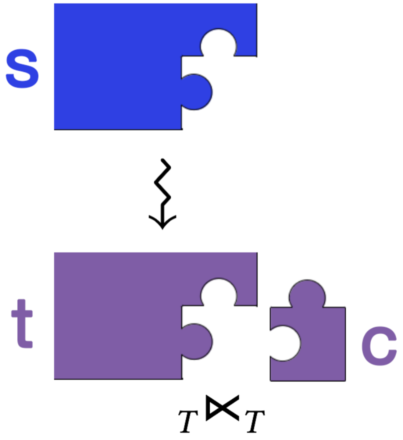
Problem this research addresses

To understand if Theorem is correct...

Pilsner	source-target PILS relation
CompComp	interaction semantics
Multi-lang	source-target multi-language

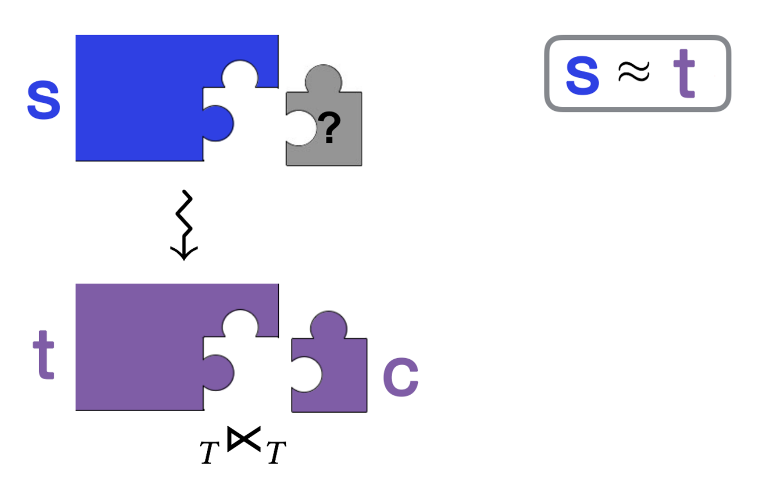
Is there a generic CCC theorem?

Theorem (CCC)

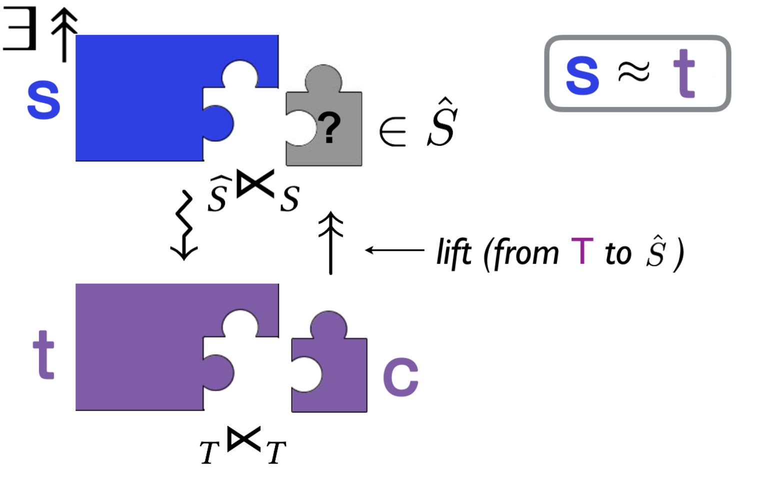


$\boxed{s \approx t}$

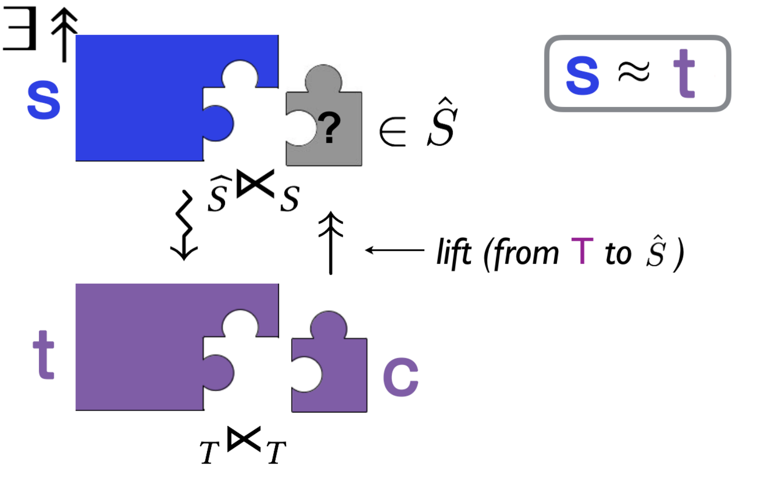
Theorem (CCC)



Theorem (CCC)


∃ ↟
s
?
$\in \hat{S}$
$s \approx t$
$\hat{S} \ltimes S$
lift (from T to $\hat{S}$)
t
c
$T \ltimes T$

Theorem (CCC)



$$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}.ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \uparrow\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

Theorem (CCC)

$$\exists \uparrow .\mathsf{s} \rightsquigarrow \mathsf{t}.\mathit{ok}_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \uparrow\!\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

Instantiated by particular formalisms:

- $\mathit{ok}_{\ltimes}$ — determines what is linkable.
- Source-like linking medium $\widehat{S}$.
- $\uparrow$ — lift from target to $\widehat{S}$.

Using CCC to understand results

Theorem (CCC: Pilsner)

$$\exists \Uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \Uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

- $ok_{\ltimes}$ — c must be PILS-related to a c'.

Using CCC to understand results

Theorem (CCC: Pilsner)

$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \Longrightarrow \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c must be PILS-related to a c'.
- Linking medium $\widehat{S}$ is source language.

Using CCC to understand results

Theorem (CCC: Pilsner)

$\exists \Uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}). \Uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c must be PILS-related to a c'.
- Linking medium $\widehat{S}$ is source language.
- $\Uparrow$ — lifts c to c' in source language.

Using CCC to understand results

Theorem (CCC: Pilsner)

$\exists \Uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}.\mathit{ok}_{\ltimes}(\mathsf{c}, \mathsf{t}). \Uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $\mathit{ok}_{\ltimes}$ — c must be PILS-related to a c'.
- Linking medium $\widehat{S}$ is source language.
- $\Uparrow$ — lifts c to c' in source language.

Weakness: $\mathit{ok}_{\ltimes}$ tells us we can only link with terms relatable to source.

Using CCC to understand results

Theorem (CCC: Multi-language)

$\exists \Uparrow . \mathsf{s} \rightsquigarrow \mathsf{t} . ok_{\ltimes}(\mathsf{c}, \mathsf{t}) . \Uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c is any well-typed target code.

Using CCC to understand results

Theorem (CCC: Multi-language)

$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}). \uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c is any well-typed target code.
- Linking medium $\widehat{S}$ is ST multi-language.

Using CCC to understand results

Theorem (CCC: Multi-language)

$\exists \Uparrow.\mathsf{s} \rightsquigarrow \mathsf{t}.ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \Uparrow\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c is any well-typed target code.
- Linking medium $\widehat{S}$ is ST multi-language.
- $\Uparrow$ — embeds c into multi-language.

Using CCC to understand results

Theorem (CCC: Multi-language)

$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t} . ok_{\ltimes}(\mathsf{c}, \mathsf{t}) . \uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c is any well-typed target code.
- Linking medium $\widehat{S}$ is ST multi-language.
- $\uparrow$ — embeds c into multi-language.

Weakness: linking medium is hard to create, formalize, and understand.

Using CCC to understand results

Theorem (CCC: CompComp)

$$\exists\, \Uparrow. \mathsf{s} \rightsquigarrow \mathsf{t}.\, ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \Uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

- $ok_{\ltimes}$ — c obeys interaction semantics.

Using CCC to understand results

Theorem (CCC: CompComp)

$$\exists \Uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}). \Uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

- $ok_{\ltimes}$ — c obeys interaction semantics.
- Linking medium $\widehat{S}$ described in Coq.

Using CCC to understand results

Theorem (CCC: CompComp)

$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}.ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \uparrow\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c obeys interaction semantics.
- Linking medium $\widehat{S}$ described in Coq.
- $\uparrow$ — embeds c in *semantic* "multi-lang".

Using CCC to understand results

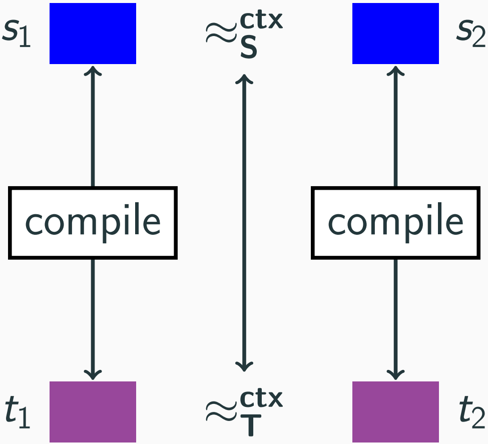
Theorem (CCC: CompComp)

$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}). \uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c obeys interaction semantics.
- Linking medium $\widehat{S}$ described in Coq.
- $\uparrow$ — embeds c in *semantic* "multi-lang".

Weakness: like multi-lang, $\widehat{S}$ hard to understand, may need to change.

Fully abstract compilation



FAbs compilers & back-translation

Theorem (CCC: Fabs w/ back-trans)

$$\exists \Uparrow . \mathsf{s} \rightsquigarrow \mathsf{t} . ok_{\ltimes}(\mathsf{c}, \mathsf{t}) . \Uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

- $ok_{\ltimes}$ — c is target code of translation type.

FAbs compilers & back-translation

Theorem (CCC: Fabs w/ back-trans)

$$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}). \uparrow \mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \Longrightarrow \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

- $ok_{\ltimes}$ — c is target code of translation type.
- Linking medium $\widehat{S}$ is source language.

FAbs compilers & back-translation

Theorem (CCC: Fabs w/ back-trans)

$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}.ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \uparrow\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c is target code of translation type.
- Linking medium $\widehat{S}$ is source language.
- $\uparrow$ — back-translates c to source.

FAbs compilers & back-translation

Theorem (CCC: Fabs w/ back-trans)

$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t} . ok_{\ltimes}(\mathsf{c}, \mathsf{t}).\ \uparrow\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c is target code of translation type.
- Linking medium $\widehat{S}$ is source language.
- $\uparrow$ — back-translates c to source.

Weakness: building fully abstract compilers is hard.

FAbs compilers & back-translation

Theorem (CCC: Fabs w/ back-trans)

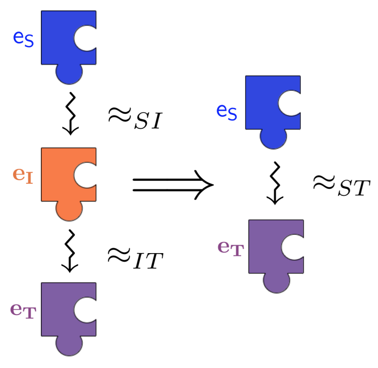
$\exists \Uparrow. \mathsf{s} \rightsquigarrow \mathsf{t}. ok_{\ltimes}(\mathsf{c}, \mathsf{t}). \Uparrow\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$

- $ok_{\ltimes}$ — c is target code of translation type.
- Linking medium $\widehat{S}$ is source language.
- $\Uparrow$ — back-translates c to source.

Strength: linking medium is just source, lift enables linking with code of arbitrary provenance.

Aside: vertical compositionality

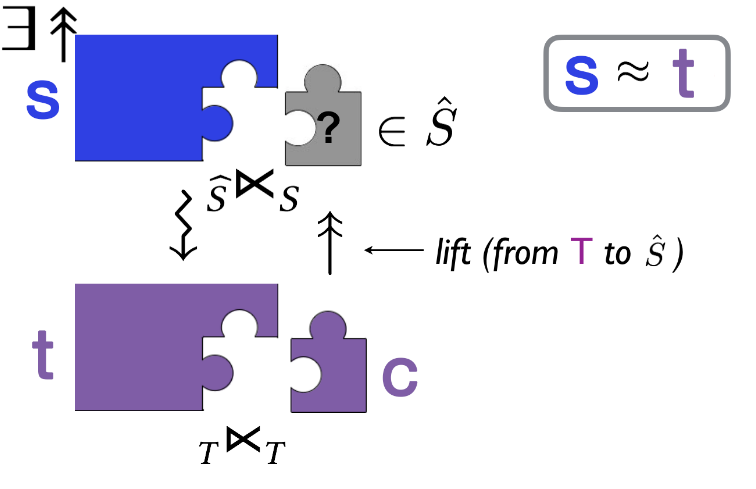
Fully abstract + CCC = Vertical compositionality



Requires back-translations, i.e., [New, Bowman, Ahmed, ICFP 2016] or [Devriese, Patrignani, Piessens, POPL 2016]

Recap: what CCC gives us

$ok_{\ltimes}(c, \cdot)$, $\Uparrow$, and $\widehat{S}$ help us understand results.



Takeaways

Theorem (CCC)

$$\exists \uparrow . \mathsf{s} \rightsquigarrow \mathsf{t}.ok_{\ltimes}(\mathsf{c},\mathsf{t}).\ \uparrow\mathsf{c} \ltimes \mathsf{s} \Downarrow \mathcal{O} \implies \mathsf{c} \ltimes \mathsf{t} \Downarrow \mathcal{O}$$

- Secure compilation **needs** definition of compositional compiler correctness.
- Fully abstract compilers $\implies$ vertical compositionality for free!