Interpolation
and
Proof Reduction

Georg Weissenbacher
& R. Bloem, S. Malik, M. Schlaipfer
ExCAPE 2014
Motivation: Circuit Synthesis

- designing ICs is hard
Motivation: Circuit Synthesis

- designing ICs is hard
- what if we could leave “holes” (black boxes)
Motivation: Circuit Synthesis

- designing ICs is hard
- what if we could leave “holes” (black boxes)
- specify behaviour using input/output relation
Motivation: Circuit Synthesis

- designing ICs is hard
- what if we could leave “holes” (black boxes)
- specify behaviour using input/output relation
- then let a program fill in the blanks
Jiang, Lin, Hung ICCAD’09

Given: quantifier-free (propositional) specification

\[ R : \text{Inputs } \times \text{Output } \rightarrow \mathbb{B} \]

Wanted: functional implementation \( I : \text{Inputs } \rightarrow \mathbb{B} \)
Given: quantifier-free (propositional) specification $R: \text{Inputs} \times \text{Output} \rightarrow \mathbb{B}$

Wanted: functional implementation $I: \text{Inputs} \rightarrow \mathbb{B}$

same setting as in logic synthesis

By reduction to [Jiang, Lin, Hung ICCAD’09]

more than one parameter by co-factoring, iterative solving
Synthesis from Relational Specifications

- Jiang, Lin, Hung ICCAD’09
  - Given: quantifier-free (propositional) specification
    \[ R : \text{Inputs} \times \text{Output} \rightarrow \mathbb{B} \]
  - Wanted: functional implementation \( I : \text{Inputs} \rightarrow \mathbb{B} \)
  - same setting as in logic synthesis

- Hofferek, Bloem MEMOCODE’11; Hofferek et al. FMCAD’13
  - Given: circuit specification with parameter \( c \):
    \[ \forall \vec{i} : \text{Inputs} \exists c : \mathbb{B} \forall \vec{o} : \text{Outputs} . R(\vec{i}, c, \vec{o}) \]
  - Wanted: functional implementation \( I : \text{Inputs} \rightarrow \mathbb{B} \)
Synthesis from Relational Specifications

- Jiang, Lin, Hung ICCAD’09
  - Given: quantifier-free (propositional) specification
    \[ R : \text{Inputs} \times \text{Output} \rightarrow \mathbb{B} \]
  - Wanted: functional implementation \( I : \text{Inputs} \rightarrow \mathbb{B} \)
  - same setting as in logic synthesis

- Hofferek, Bloem MEMOCODE’11; Hofferek et al. FMCAD’13
  - Given: circuit specification with parameter \( c \):
    \[ \forall \vec{i} : \text{Inputs} \ \exists c : \mathbb{B} \ \forall \vec{o} : \text{Outputs} . \ R(\vec{i}, c, \vec{o}) \]
  - Wanted: functional implementation \( I : \text{Inputs} \rightarrow \mathbb{B} \)
  - by reduction to [Jiang, Lin, Hung ICCAD’09]
  - more than one parameter by co-factoring, iterative solving
Interpolant *separates* “onset” from “offset”
What is a Craig-Robinson Interpolant?

Craig-Robinson interpolant for inconsistent (first-order) conjunction $A \land B$:
- $A \Rightarrow I$ and $I \land B$ inconsistent
- all non-logical symbols in $I$ occur in $A$ as well as in $B$
$\neg R(\vec{i}, 0)$  onset only

$\neg R(\vec{i}, 1)$  offset only
\[ \neg R(\vec{i}, 0) \quad \land \quad \neg R(\vec{i}, 1) \]

- onset only
- offset only
Interpolants and Synthesis

\[ \neg R(\vec{i}, 0) \quad \land \quad \neg R(\vec{i}, 1) \]

onset only

offset only
State of the Art: Interpolants from Refutation Proofs

\[ A \quad B \]

\[ \perp \]
State of the Art: Interpolants from Refutation Proofs
State of the Art: Interpolants from Refutation Proofs

\[ A \vdash B \]

\[ A_0, C_1[l_1], B_1, A_1, C_2[l_2] \]

\[ \perp \]
State of the Art: Interpolants from Refutation Proofs

\[ \bot \]

G. Weissenbacher (TU Wien)
Interpolation Systems

- map proof nodes to *partial interpolants*
- root node is an interpolant for $A \land B$
- Intuition:
  - eliminate *local* literals (そうでない) as in proof
  - keep *shared* literals (すぐ)
Interpolation Systems

- map proof nodes to *partial interpolants*
- root node is an interpolant for $A \land B$
- Intuition:
  - eliminate *local* literals (🌈, 🌈) as in proof
  - keep *shared* literals (🌈)
- In synthesis setting, *all* inputs are shared
- Symbols in interpolant determined by *unsatisfiable core*
Labelled Interpolation Systems [D’Silva et al. VMCAI’10]

- conceived to fine-tune logical strength of interpolants
Labelled Interpolation Systems [D’Silva et al. VMCAI’10]

- conceived to fine-tune logical strength of interpolants
- literals can be labelled individually (\(\hat{}\), \(\hat{}\), \(\hat{}\))
  - allows us to treat literals as if they were local [D’Silva ESOP’10]

\[
\begin{align*}
A(i, j) & \quad \quad B(i, j) \\
\downarrow & \\
\bot
\end{align*}
\]
Labelled Interpolation Systems [D’Silva et al. VMCAI’10]

- conceived to fine-tune logical strength of interpolants
- literals can be labelled *individually* (🎨, 🎨, 🎨)
  - allows us to treat literals *as if they were local* [D’Silva ESOP’10]

\[
\begin{align*}
A(i, j) & \quad B(i, j) \\
\exists i . A(i, j) & \quad \exists i . B(i, j)
\end{align*}
\]

\[\bot\]
Labelled Interpolation Systems [D’Silva et al. VMCAI’10]

- conceived to fine-tune logical strength of interpolants
- literals can be labelled *individually* (__, __, __)
  - allows us to treat literals as if they were local [D’Silva ESOP’10]

\[
\begin{array}{c}
A(i, j) \quad B(i, j) \\
\hline
A(j) \quad B(j) \\
\hline
\quad \Downarrow \quad \lfloor I(j) \rfloor
\end{array}
\]
Labelled Interpolation: Example

\[ x_0 \bar{x}_1 \quad [0] \quad x_1 \quad x_2 \quad [0] \]

\[ x_0 \quad x_2 \quad [0] \quad x_1 \bar{x}_2 \quad [1] \]

\[ x_0 \quad x_1 \quad [x_2] \quad \bar{x}_1 \quad [1] \]

\[ \bar{x}_0 \quad [0] \quad x_0 \quad [x_2] \]

\[ \square \quad [x_2] \]
Labelled Interpolation: Example

\[
\begin{array}{c}
\text{x}_0 \quad \overline{\text{x}}_1 \quad [0] \\
\overline{\text{x}}_0 \quad [0] \quad \text{x}_0 \\
\text{x}_1 \quad \text{x}_2 \quad [0] \\
\text{x}_1 \quad \overline{\text{x}}_2 \quad [1] \\
\end{array}
\]
Labelled Interpolation: Example

Literals in interpolant dictated by proof (core and structure)
Numerous techniques for reduction of proof size:
- Simone et al. HVC’10; Bar-Ilan et al. STTT’11; Gupta ATVA’12; …
- Most can be viewed as (generalised) clause sub-sumption
Smaller Interpolants via Proof Reduction?

\[ x_0 \ \overline{x}_1 \quad [0] \quad x_1 \ x_2 \quad [0] \]
\[ x_0 \ x_2 \quad [0] \quad x_1 \ \overline{x}_2 \quad [1] \]
\[ x_0 \ x_1 \quad [x_2] \quad \overline{x}_1 \quad [1] \]
\[ \overline{x}_0 \quad [0] \quad x_0 \quad [x_2] \]
\[ \Box \quad [x_2] \]
Smaller Interpolants via Proof Reduction?

\[ x_0 \quad x_1 \quad \overline{x}_1 \quad [0] \quad x_1 \quad x_2 \quad [0] \]

\[ x_1 \quad x_0 \quad x_2 \quad [0] \quad x_1 \quad \overline{x}_2 \quad [1] \]

\[ \overline{x}_0 \quad [0] \quad x_0 \quad [x_2] \]

\[ \square \quad [x_2] \]
Smaller Interpolants via Proof Reduction?

\[
\begin{align*}
\neg x_0 & \land \neg x_1 [0] & \land x_1 x_2 [0] \\
\land x_1 x_2 [0] & \land x_1 \neg x_2 [1] \\
\land x_1 [x_2] & \land \neg x_1 [1] \\
\land \neg x_0 [0] & \land [x_1 \lor x_2]
\end{align*}
\]
Smaller Interpolants via Proof Reduction?

Transformation made -local literal shared ( )
Smaller Interpolants via Proof Reduction

\[ x_0 \quad \overline{x}_1 \quad [0] \quad x_1 \quad x_2 \quad [0] \]

\[ x_1 \quad x_0 \quad x_2 \quad [0] \quad x_1 \quad \overline{x}_2 \quad [1] \]

\[ x_0 \quad x_1 \quad [x_2] \quad \overline{x}_1 \quad [1] \]

\[ \overline{x}_0 \quad [0] \quad x_0 \quad [x_2] \]

\[ \square \quad [x_2] \]
Our current work:

- *Meet-over-all-paths* analysis of refutation (linear time)
- Identifies constraints for sub-sumption
  - avoids reduction in previous example
- 1%-5% reduction of # variables (HWMCC; SATC/MUS benchmarks)
- 10%-20% reduction of proof size
Our current work:

- *Meet-over-all-paths* analysis of refutation (linear time)
- Identifies constraints for sub-sumption
  - avoids reduction in previous example
- 1%-5% reduction of # variables (HWMCC; SATC/MUS benchmarks)
- 10%-20% reduction of proof size

Future work:

- More aggressive “static” analysis of proofs
- Heuristics for identifying beneficial sub-sumptions
- Eliminate *specific* (user-defined) symbols
- Combine with recent MUS-techniques (Marques-Silva’s work)
Application deadline: **April 19** (http://satsmt2014.forsyte.at)

Generous NSF travel grants for US students