EXPLAIN:
A Tool for Performing Abductive Inference

Isil Dillig
MSR Cambridge
Abduction: Opposite of deduction
What is Abduction?

- **Abduction**: Opposite of deduction
- **Deduction**: Infers valid conclusion from premises
Abduction: Opposite of deduction

Deduction: Infers valid conclusion from premises

Abduction: Infers missing premise to explain a given conclusion
What is Abduction?

- **Abduction**: Opposite of deduction
- **Deduction**: Infers valid conclusion from premises
- **Abduction**: Infers missing premise to explain a given conclusion

Given known facts $\Gamma$ and desired outcome $\phi$, abductive inference finds “simple” explanatory hypothesis $\psi$ such that

$$\Gamma \land \psi \models \phi \quad \text{and} \quad \text{SAT}(\Gamma \land \psi)$$
Simple Example

Facts: “If it rains, then it is wet and cloudy”, “If it is wet, then it is slippery”:
$$R \Rightarrow W \land C \land W \Rightarrow S$$

Conclusion: “It is cloudy and slippery”, i.e.,
$$C \land S$$

Abductive explanation: $R$, i.e.,”It is rainy”
Simple Example

- Facts: “If it rains, then it is wet and cloudy”, “If it is wet, then it is slippery”:
  \[ R \Rightarrow W \land C \land W \Rightarrow S \]

- Conclusion: “It is cloudy and slippery”, i.e., \[ C \land S \]
Simple Example

- Facts: “If it rains, then it is wet and cloudy”, “If it is wet, then it is slippery”:
  \[ R \Rightarrow W \land C \land W \Rightarrow S \]

- Conclusion: “It is cloudy and slippery”, i.e., \( C \land S \)

- Abductive explanation: \( R \), i.e., “It is rainy”
int x = 0;
int y = 0;

while(x < n)
{
    x = x+1;
    y = y+2;
}

assert( x + y >= 3*n);
Suppose we know $x \geq n$

- e.g., from loop termination condition
Suppose we know $x \geq n$

- e.g., from loop termination condition

Desired conclusion $x + y \geq 3n$

- property we want to prove
Suppose we know $x \geq n$

- e.g., from loop termination condition

Desired conclusion $x + y \geq 3n$

- property we want to prove

Abductive explanation: $y \geq 2x$

- corresponds to missing loop invariant
In general, the abduction problem $\Gamma \land ? \models \phi$ has infinitely many solutions.
Properties of Desired Solutions

- In general, the abduction problem $\Gamma \land ? \models \phi$ has infinitely many solutions.

- **Trivial solution:** $\phi$, but not useful because it does not take into account what we know.
In general, the abduction problem $\Gamma \land ? \models \phi$ has infinitely many solutions.

**Trivial solution:** $\phi$, but not useful because it does not take into account what we know.

So, what kind of solutions do we want to compute?
Which Abductive Explanations Are Good?

Guiding Principle:
Occam’s Razor

If there are multiple competing hypotheses, select the one that makes fewest assumptions.

Generality: If explanation A is logically weaker than explanation B, always prefer A.

Simplicity: Not clear-cut, but we use number of variables.

This simplicity criterion makes sense in verification because we want proof subgoals to be local and refer to few variables.
Guiding Principle: Occam’s Razor

- If there are multiple competing hypotheses, select the one that makes fewest assumptions
If there are multiple competing hypotheses, select the one that makes fewest assumptions

Generality: If explanation $A$ is logically weaker than explanation $B$, always prefer $A$
Which Abductive Explanations Are Good?

Guiding Principle: Occam’s Razor

- If there are multiple competing hypotheses, select the one that makes fewest assumptions.

- **Generality**: If explanation $A$ is logically weaker than explanation $B$, always prefer $A$.

- **Simplicity**: Not clear-cut, but we use number of variables.
Which Abductive Explanations Are Good?

Guiding Principle: Occam’s Razor

- If there are multiple competing hypotheses, select the one that makes fewest assumptions

- **Generality:** If explanation $A$ is logically weaker than explanation $B$, always prefer $A$

- **Simplicity:** Not clear-cut, but we use number of variables

- This simplicity criterion makes sense in verification because we want proof subgoals to be local and refer to few variables
EXPLAIN’s Abduction Algorithm

- EXPLAIN computes a **logically weakest** solution with **fewest** variables to abduction problems in Presburger arithmetic.
EXPLAIN’s Abduction Algorithm

- EXPLAIN computes a logically weakest solution with fewest variables to abduction problems in Presburger arithmetic.

- Given premises $I$ and desired conclusion $\phi$:
EXPLAIN’s Abduction Algorithm

- EXPLAIN computes a **logically weakest** solution with **fewest** variables to abduction problems in Presburger arithmetic.

- Given premises $I$ and desired conclusion $\phi$:

1. Compute an MSA of $I \Rightarrow \phi$ consistent with $I$


\[
\text{abduce}(I, \phi) \quad \{
\]

\[
V = \text{msa}(I \Rightarrow \phi, I)
\]

\[
\}
\]
EXPLAIN’s Abduction Algorithm

EXPLAIN computes a **logically weakest** solution with **fewest variables** to abduction problems in Presburger arithmetic

Given premises $I$ and desired conclusion $\phi$:

1. Compute an MSA of $I \Rightarrow \phi$ consistent with $I$

2. Quantify out all variables not in the MSA

```
abduce(I, \phi) { 
  V = msa(I \Rightarrow \phi, I) 
  \psi = QE(\forall V.(I \Rightarrow \phi)) 
}
```
EXPLAIN’s Abduction Algorithm

EXPLAIN computes a logically weakest solution with fewest variables to abduction problems in Presburger arithmetic.

Given premises $I$ and desired conclusion $\phi$:

1. Compute an MSA of $I \Rightarrow \phi$ consistent with $I$

2. Quantify out all variables not in the MSA

3. Remove subparts implied or contradicted by premises

```
abduce(I, \phi) \{
V = msa(I \Rightarrow \phi, I)

\psi = QE(\forall \overline{V}.(I \Rightarrow \phi))

\psi' = simplify(\psi, I)
\}
```
EXPLAIN’s Abduction Algorithm

- EXPLAIN computes a **logically weakest solution with fewest variables** to abduction problems in Presburger arithmetic.

- Given premises \( I \) and desired conclusion \( \phi \):
  1. Compute an MSA of \( I \Rightarrow \phi \) consistent with \( I \)
  2. Quantify out all variables not in the MSA
  3. Remove subparts implied or contradicted by premises

**abduce**\( (I, \phi) \)  

\[
V = \text{msa}(I \Rightarrow \phi, I) \\
\psi = \text{QE}(\forall V . (I \Rightarrow \phi)) \\
\psi' = \text{simplify}(\psi, I) \\
\text{return } \psi'
\]
Useful technique to add to our bag of tricks; lots of applications!
Abduction in Program Analysis

Useful technique to add to our bag of tricks; lots of applications!

- Loop invariant generation
Abduction in Program Analysis

Useful technique to add to our bag of tricks; lots of applications!

- Loop invariant generation
- Synthesis of compositional program proofs
Useful technique to add to our bag of tricks; lots of applications!

- Loop invariant generation
- Synthesis of compositional program proofs
- Inference of missing library specifications
Useful technique to add to our bag of tricks; lots of applications!

- Loop invariant generation
- Synthesis of compositional program proofs
- Inference of missing library specifications
- Explaining static analysis warnings to programmers
Useful technique to add to our bag of tricks; lots of applications!

- Loop invariant generation
- Synthesis of compositional program proofs
- Inference of missing library specifications
- Explaining static analysis warnings to programmers
- Modular analysis using separation logic
EXPLAIN is implemented in Mistral SMT solver and is available from:
http://www.cs.wm.edu/~tdillig/mistral
EXPLAIN is implemented in Mistral SMT solver and is available from:
http://www.cs.wm.edu/~tdillig/mistral

- The tool paper describes algorithm in more detail and presents usage examples.
EXPLAIN is implemented in Mistral SMT solver and is available from:
http://www.cs.wm.edu/~tdillig/mistral

The tool paper describes algorithm in more detail and presents usage examples

Try it out!