

# Feedback-Driven Semi-Supervised Synthesis of Program Transformations

Xiang Gao, Shraddha Barke, Arjun Radhakrishna, Gustavo Soares, Sumit Gulwani, Alan  
Leung, Nachiappan Nagappan, Ashish Tiwari



# Learning Program transformations from examples

POSTED ON NOV 6, 2018 TO DEVELOPER TOOLS, ML APPLICATIONS

## Getafix: How Facebook tools learn to fix bugs automatically

```
cat.h0();  
    { if (h0 == null)  
      return;  
      h0.h0();  
    }  
  
{ cat.h0();  
  }  
  { if (cat == null)  
    return;  
    cat.h0();  
  }  
  errorVar: cat  
  
1 { cat.sleep();  
  }  
  { if (cat == null)  
    return;  
    cat.sleep();  
  }  
  errorVar: cat  
  
1 { cat.meow(volume);  
  }  
  { if (cat == null)  
    return;  
    cat.meow(volume);  
  }  
  errorVar: cat  
  
{ dog.drink(h0);  
  }  
  { if (dog == null)  
    return;  
    dog.drink(h0);  
  }  
  errorVar: dog
```

Merge edits which check `cat` for null before calling `cat.h0()` (either `sleep()` or `meow()`)

By Satish Chandra, Johannes Bader, Eric Lippert, Andrew Scott

Bader et al. [OOPSLA'19]

## Making repeated edits easier with IntelliCode suggestions

Peter  
August 6th, 2020

What if your developer tools could track your edits and learn while you are making changes? What if they could offer to do remaining edits for you?

Your repeated edit experience is now enhanced by [IntelliCode suggestions](#) in [Visual Studio 2019 16.7](#). IntelliCode spots repetitions and **suggests** other places in your code where you could apply that same change.

```
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74
```

Milner et al. [OOPSLA'19]

Overfit to given examples

```
RepositoryModelController.cs  X
CodeSmarts.Services.Core  CodeSmarts.Services.Core.Controllers.RepositoryModelController  BuildModelOutputDetailsResponse(Model model, ModelOutput output, string user
81  //private IRepositoryModelManager RepositoryModelManager { get; }
82
83  2 references | Gustavo Soares, 211 days ago | 2 authors, 2 changes | 0 exceptions
84  private ModelOutputDetailsResponse BuildModelOutputDetailsResponse(Model model, ModelOutput output, string userId)
85  {
86      return new ModelOutputDetailsResponse
87      {
88          Model = BuildModelResponseMinimal(model),
89          Output = BuildModelOutputResponse(output, model.Id),
90          TrainingPermission = RepositoryModelManager.GetModelTrainingPermissions(model, userId),
91      };
92  }
93  /// <summary>
94  /// API endpoint to post a repository-attached model. It will provide a challenge to
95  /// the caller, who can then call again providing a response to the challenge. If
96  /// the response is satisfactory then we'll allow training to occur.
97  /// </summary>
98  /// <param name="modelCreateRequest">The model create request.</param>
99  /// <returns>An action result for the API operation.</returns>
100 [HttpPost]
101 [ProducesResponseType(typeof(ModelInputDetailsResponse), StatusCodes.Status200OK)]
```

IntelliCode suggestions  
Suggestions based on recent edits

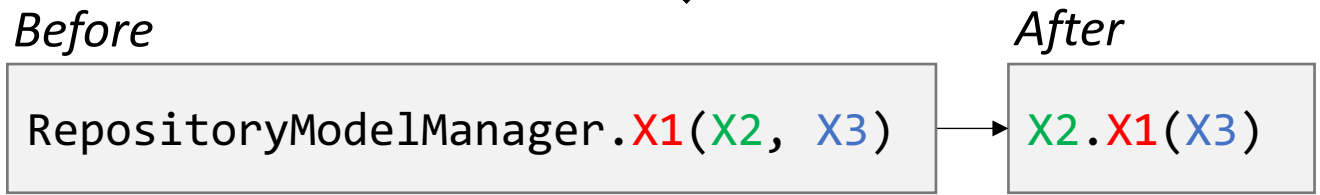


- RepositoryModelManager.GetModelTrainingPermissions(model, userId)
- + model.GetModelTrainingPermissions(userId)
  
- RepositoryModelManager.GetModelTrainingPermissions(model, currentUserId)
- + model.GetModelTrainingPermissions(currentUserId)



Blue-Pencil  
Synthesizer

Program ↓



X1.type = identifier  $\wedge$  X1.Value = "GetModelTrainingPermissions"

X2.type = identifier  $\wedge$  X2.Value = "model"

X3.type = identifier

*Before*

RepositoryModelManager.X1(X2, X3)

*After*

X2.X1(X3)

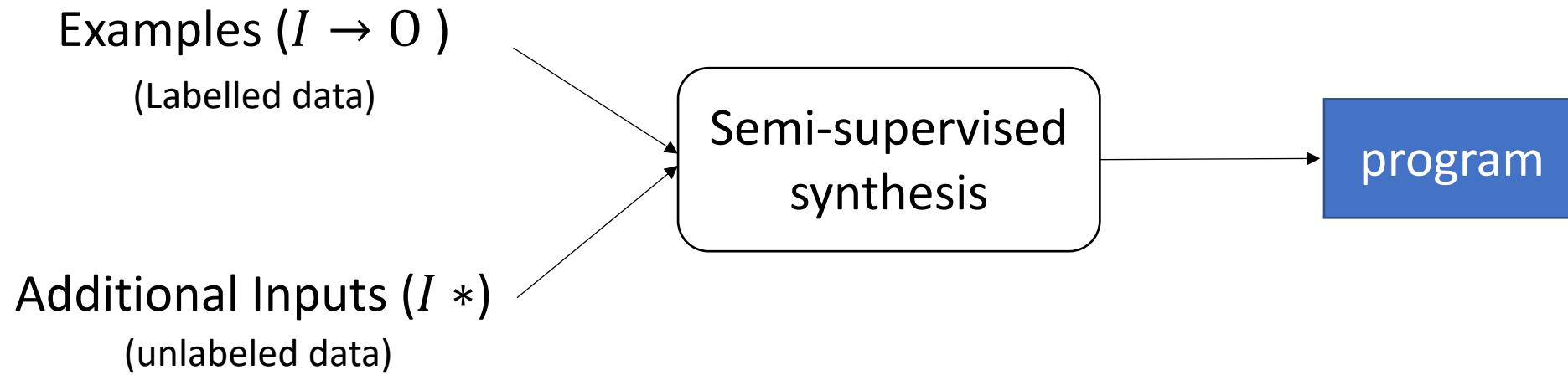
X1.type = identifier  $\wedge$  X1.Value = "GetModelTrainingPermissions"  
X2.type = identifier  $\wedge$  X2.Value = "model"  
X3.type = identifier

**Additional input:** RepositoryModelManager.IsTrainingUnderway(model, userId)

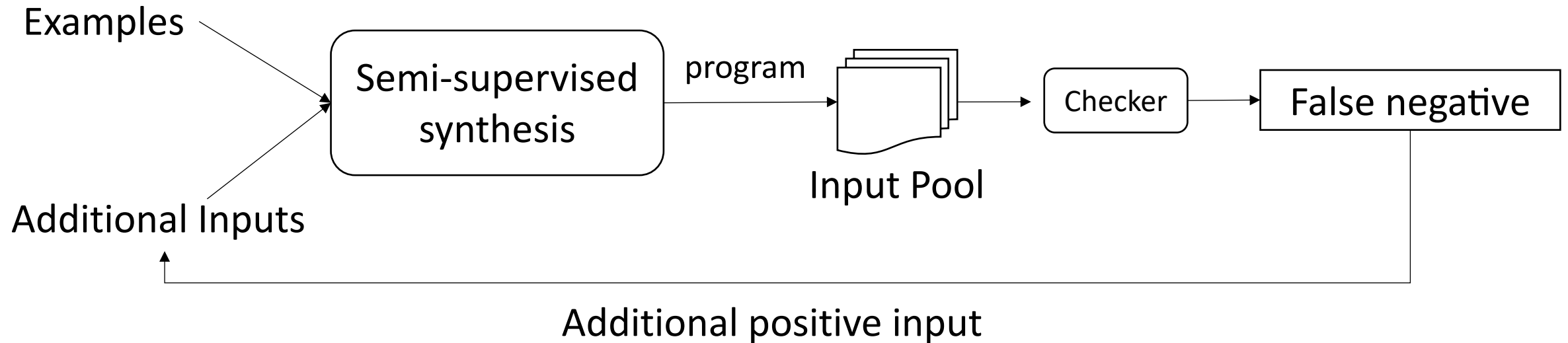
**Expected output:** model.IsTrainingUnderway(userId)

**Actual output:** 

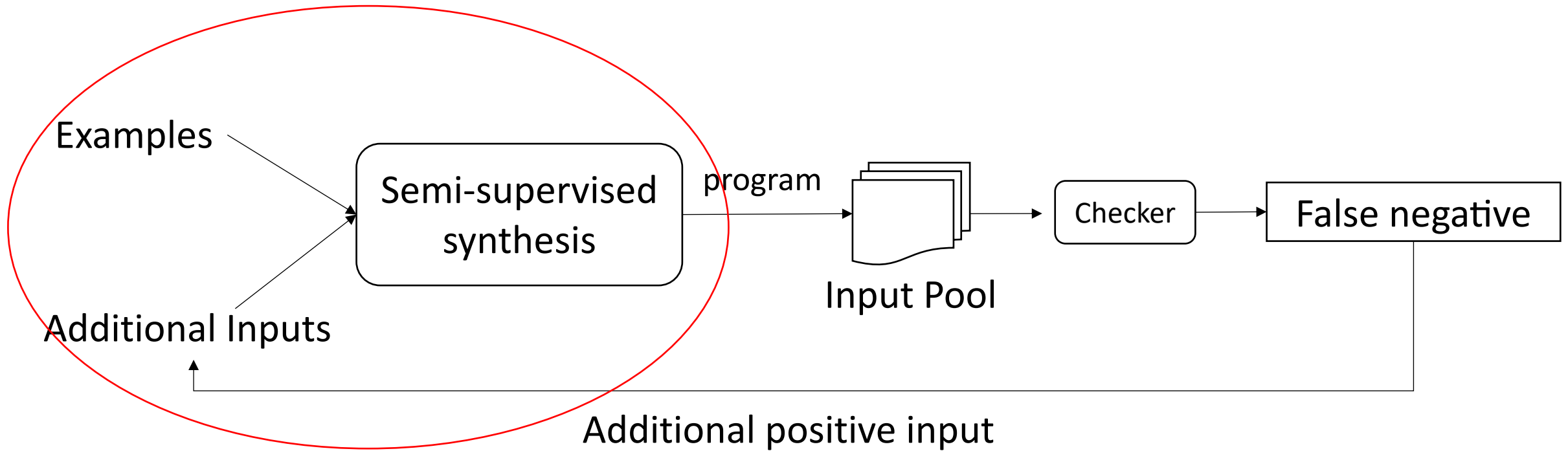
# Semi-supervised synthesis of program transformations



# Feedback Driven Semi-supervised synthesis of program transformations

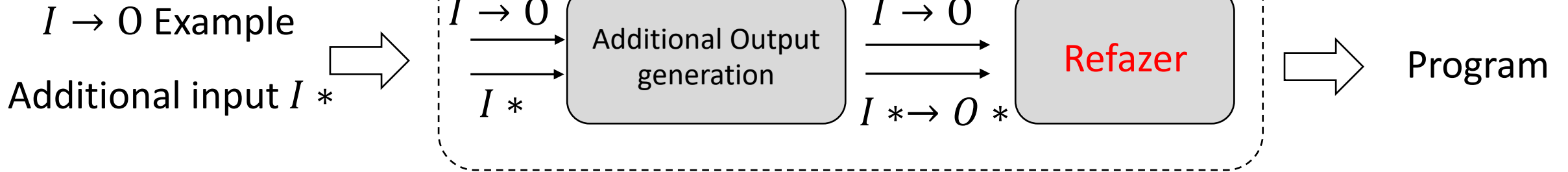


# Feedback Driven Semi-supervised synthesis of program transformations





# Semi-supervised synthesis



$I \rightarrow O$

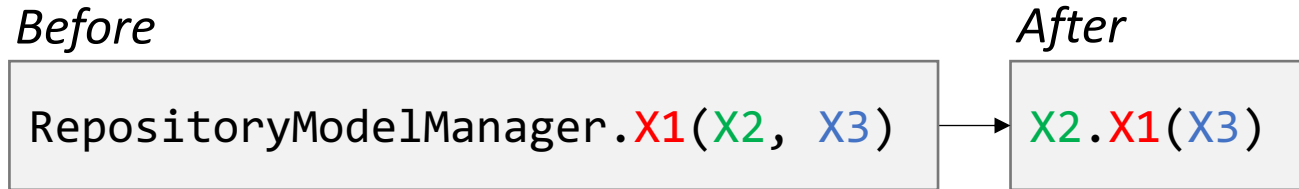
```
- RepositoryModelManager.GetModelTrainingPermissions(model, userId)  
+ model.GetModelTrainingPermissions(userId)
```

$I^* \rightarrow O^*$

```
Additional input: RepositoryModelManager.IsTrainingUnderway(model, userId)  
Additional output: model.IsTrainingUnderway(userId)
```

# Provenance analysis

Calculate a set of maps between input and output.

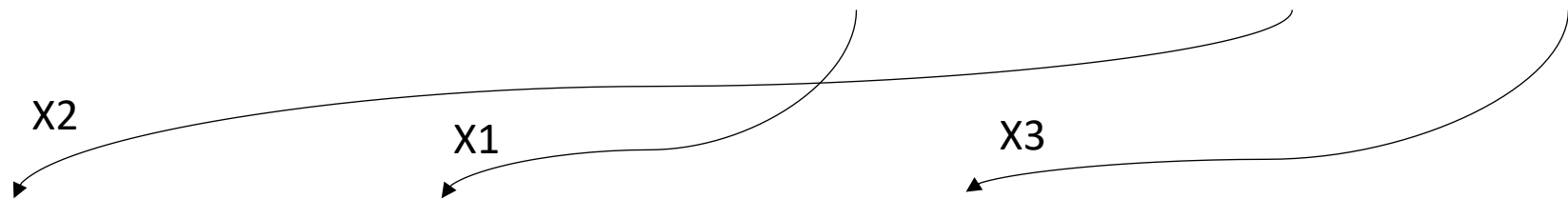


X1.type = identifier  $\wedge$  X1.Value = "GetModelTrainingPermissions"

X2.type = identifier  $\wedge$  X2.Value = "model"

X3.type = identifier

**Before:** RepositoryModelManager.GetModelTrainingPermissions(model, currentUserId)



**After:** model.GetModelTrainingPermissions(currentUserId);

# Anti-unification modulo provenance

input: RepositoryModelManager.`GetModelTrainingPermissions`(model, `currentUserId`)

additional input: RepositoryModelManager.`IsTrainingUnderway`(model, `userId`)

Anti-unification modulo  
provenance

Generalization:

RepositoryModelManager.`X1`(model, `X3`)

Input Substitution: { `X1` → `GetModelTrainingPermissions` , `X3` → `currentUserId` }

Additional input Substitution: { `X1` → `IsTrainingUnderway` , `X3` → `userId` }

Additional input Substitution:  $\{X_1 \rightarrow \text{IsTrainingUnderway}, X_3 \rightarrow \text{userId}\}$

*Before*

RepositoryModelManager.X1(X2, X3)

*After*

X2.X1(X3)

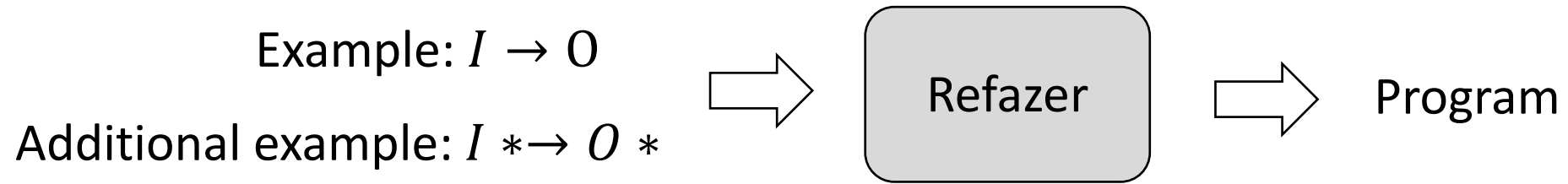
X1.type = identifier  $\wedge$  X1.Value = "GetModelTrainingPermissions"

X2.type = identifier  $\wedge$  X2.Value = "model"

X3.type = identifier

**additional input:** RepositoryModelManager.IsTrainingUnderway(model, userId)

**additional output:** model.IsTrainingUnderway(userId)



*Before*

RepositoryModelManager.X1(X2, X3)

*After*

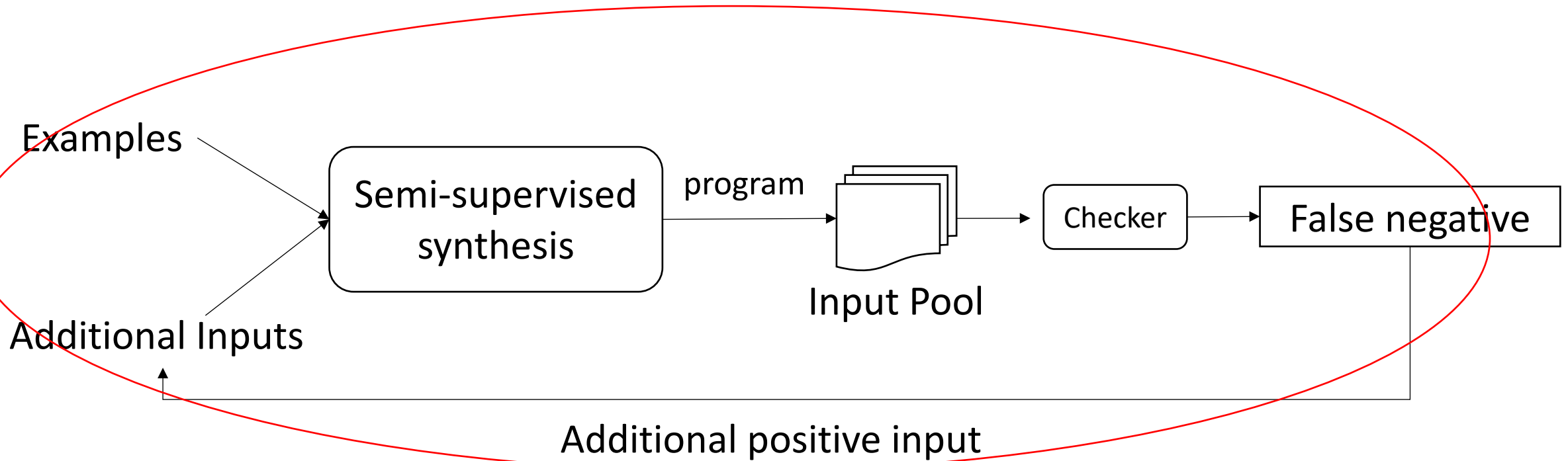
X2.X1(X3)

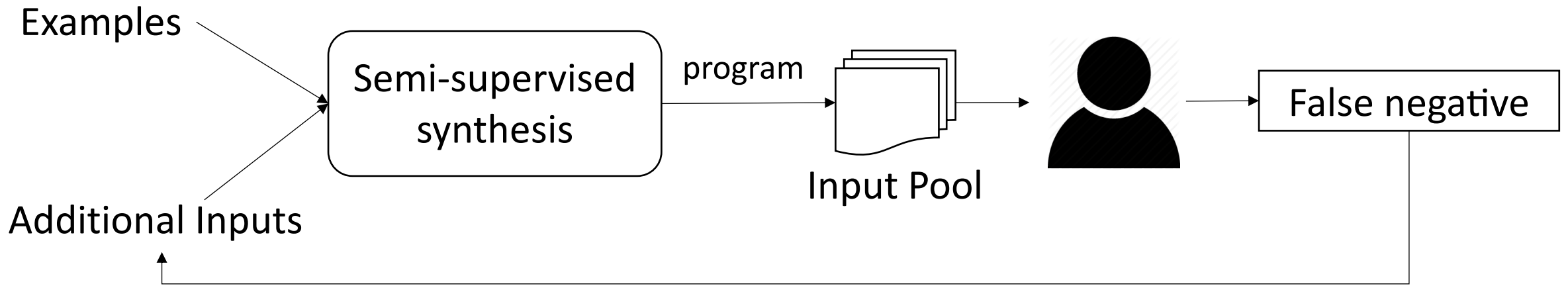
X1.type = identifier  $\wedge$  ~~X1.Value = "GetModelTrainingPermissions"~~

X2.type = identifier  $\wedge$  X2.Value = "model"

X3.type = identifier

# Feedback Driven Semi-supervised synthesis of program transformations



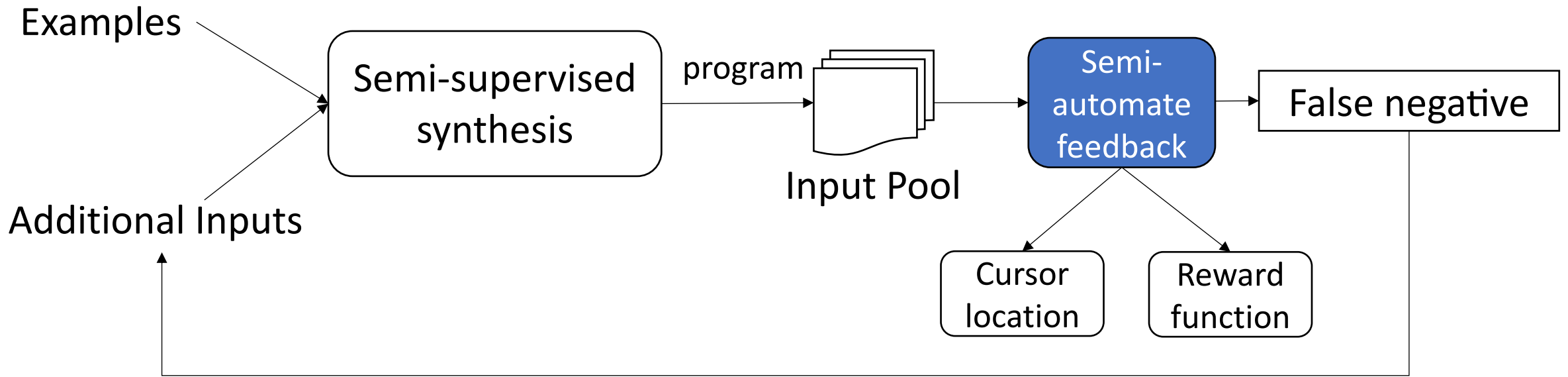


### Input pool



```
...  
var userId = HttpContext.GetCurrentUserId();  
if (string.IsNullOrEmpty(currentUserId))  
{  
    return Unauthorized();  
}  
  
if (string.IsNullOrEmpty(modelCreateRequest.RepositoryId))  
{  
    return BadRequest();  
}  
RepositoryModelManager.IsTrainingUnderway(model, userId)  
...  

```



```

...
if (string.IsNullOrEmpty(currentUserId))
{
    return Unauthorized();
}

if (string.IsNullOrEmpty(modelCreateRequest.RepositoryId))
{
    return BadRequest();
}
RepositoryModelManager.IsTrainingUnderway(model, userId);

```

RepositoryManager model.IsTrainingUnderway(userId) ✓  
 RepositoryManager.IsTrainingUnderway  
 RepositoryManager

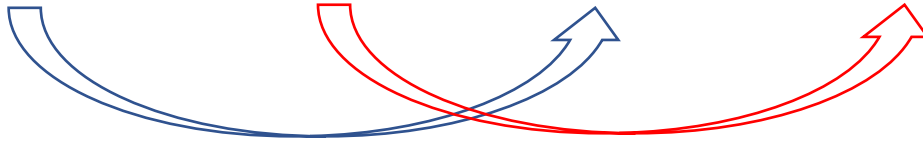


# Evaluation

- What is the effectiveness of semi-supervised synthesis in generating correct code transformations?
- Given a cursor location, what is the effectiveness of semi-automated feedback?

# Evaluation - semi-supervised synthesis

Scenario	W/O additional input		With additional input	
	Recall	Precision	Recall	Precision
1400	26.71%	100%	100%	96.01%

A diagram consisting of two curved arrows. A blue arrow starts from the 'Recall' cell (26.71%) of the 'W/O additional input' column and points to the 'Recall' cell (100%) of the 'With additional input' column. A red arrow starts from the 'Precision' cell (100%) of the 'W/O additional input' column and points to the 'Precision' cell (96.01%) of the 'With additional input' column.

Semi-supervised synthesis significantly improves the recall of baseline from 27% to 100% while retaining the high precision in generating correct suggestions.

# Evaluation - semi-automated feedback

Scenarios	Suggestion	False Positive	False Negative
295	291	1	3

Given a few past edits and one cursor location, our approach achieves around 99% precision and recall in generating correct suggestions.

# Conclusion



Existing synthesis produces overfitted programs;



Semi-supervised synthesis;



Feedback loop to generate additional inputs;



We implemented our idea as a VS plugin;



Evaluation results show we could improve the recall without loss too much precision.



**Blue-Pencil**



**PROSE**