Learning to Represent Programs with Graphs

Michael Whittaker

All code has bugs

“If debugging is the process of removing bugs, then programming must be the process of putting them in.”

—Edsger W. Dijkstra

The problem: automatically find bugs in code.

Problem: VarNaming

```
import os

for root, dirs, files in os.walk("/mydir"):
    for [redacted] in files:
        if [redacted].endswith(".txt"):
            print(os.path.join(root, [redacted]))
```

Problem: VarNaming

```
import os

for root, dirs, files in os.walk("/mydir"):
    for file in files:
        if file.endswith(".txt"):
            print(os.path.join(root, file))
```

Problem: VarNaming

Assumptions:

1. The program is written in C#, it compiles, and all type information is available
2. Only one variable name is inferred at a time; all other variables are appropriately named

Problem: VarNaming

```
import os

for [redacted], [redacted], [redacted] in os.walk("/mydir"):
    for [redacted] in [redacted]:
        if [redacted].endswith(".txt"):
            print(os.path.join([redacted], [redacted]))
```

Problem: VarMisuse

```
var clazz = classTypes["Root"].Single();
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single();
Assert.NotNull(clazz);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Problem: VarMisuse

```
var clazz = classTypes["Root"].Single();
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single();
Assert.NotNull(clazz);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Impact

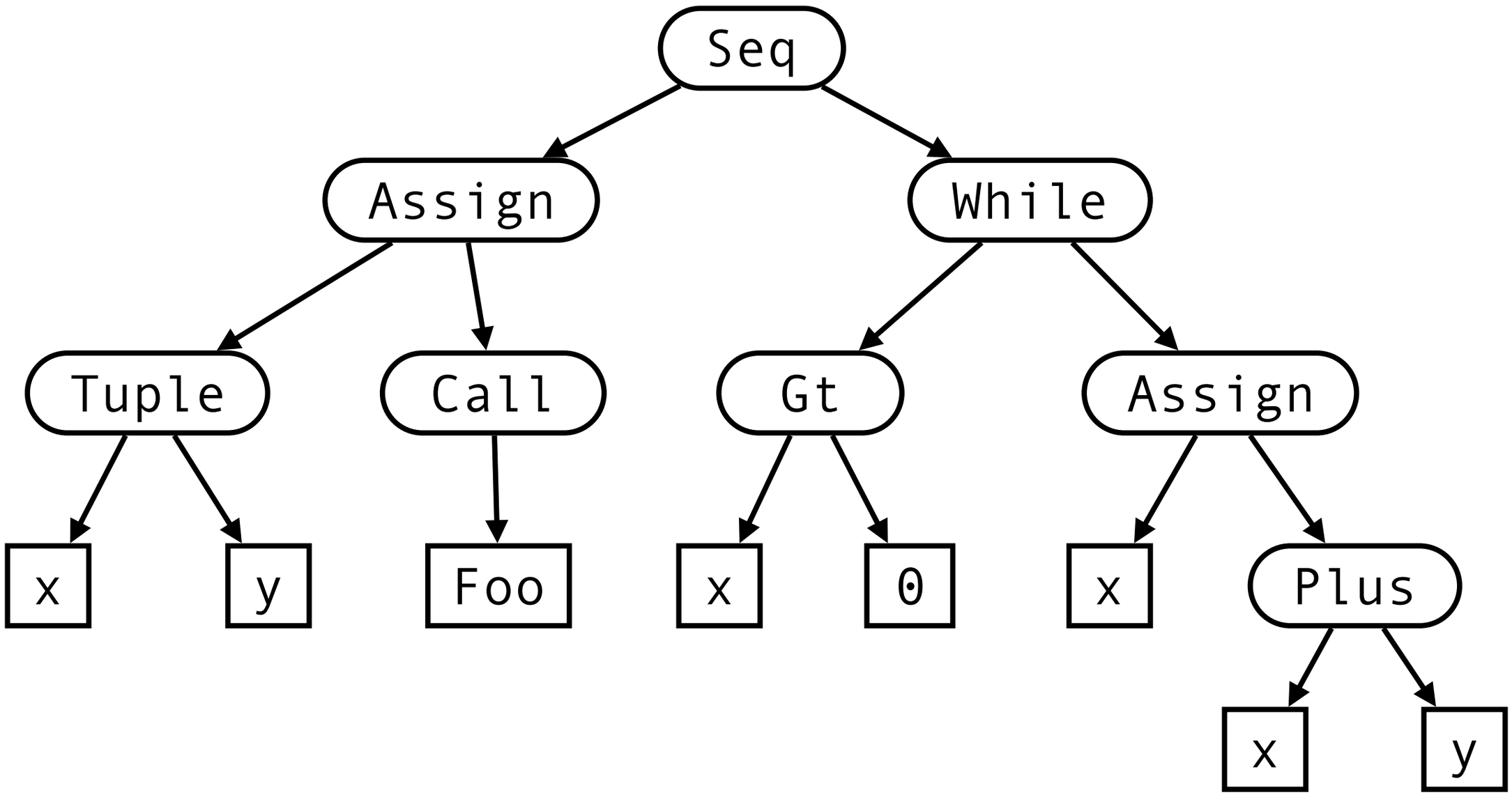
- **Industry**: these techniques can be integrated into code review, IDEs, and continuous integration to catch simple bugs.
- **Academia**: the idea of integrating semantics into program graphs is novel, but the models used are not.

Main Idea

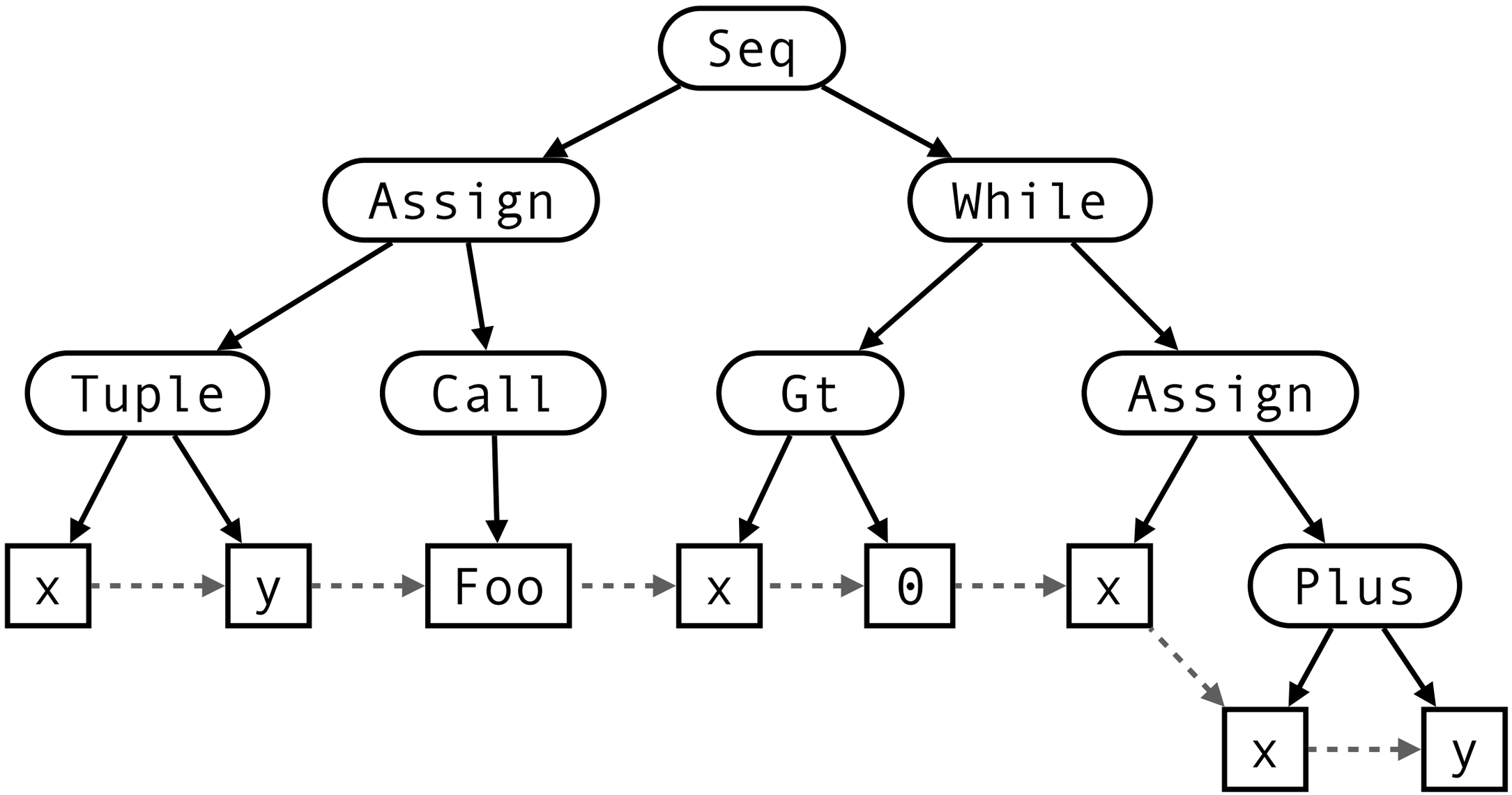
1. Represent programs as graphs
2. Add semantic edges
3. Use graph neural networks

```
(x, y) = Foo()
while (x > 0):
    x = x + y
```

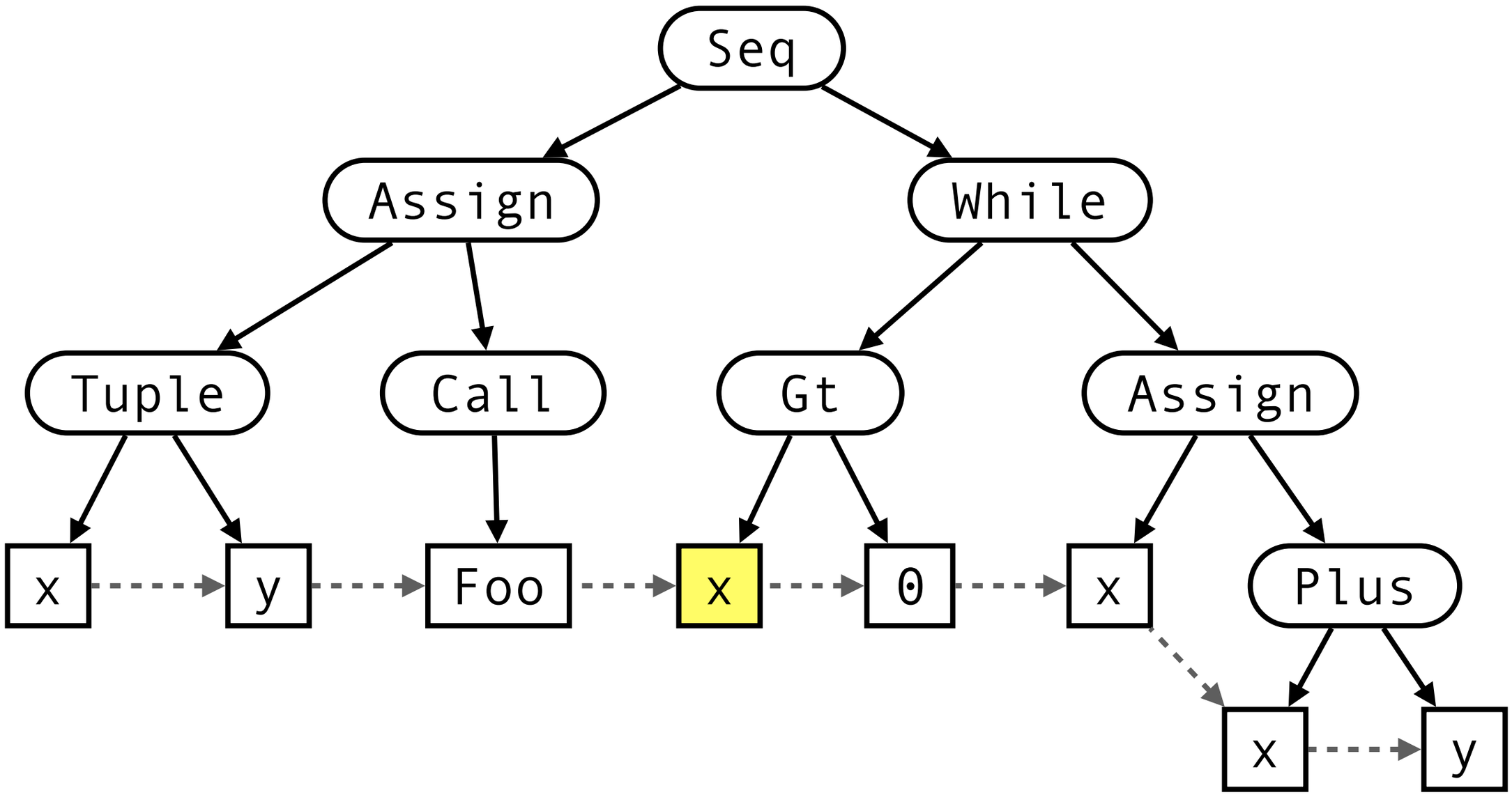
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



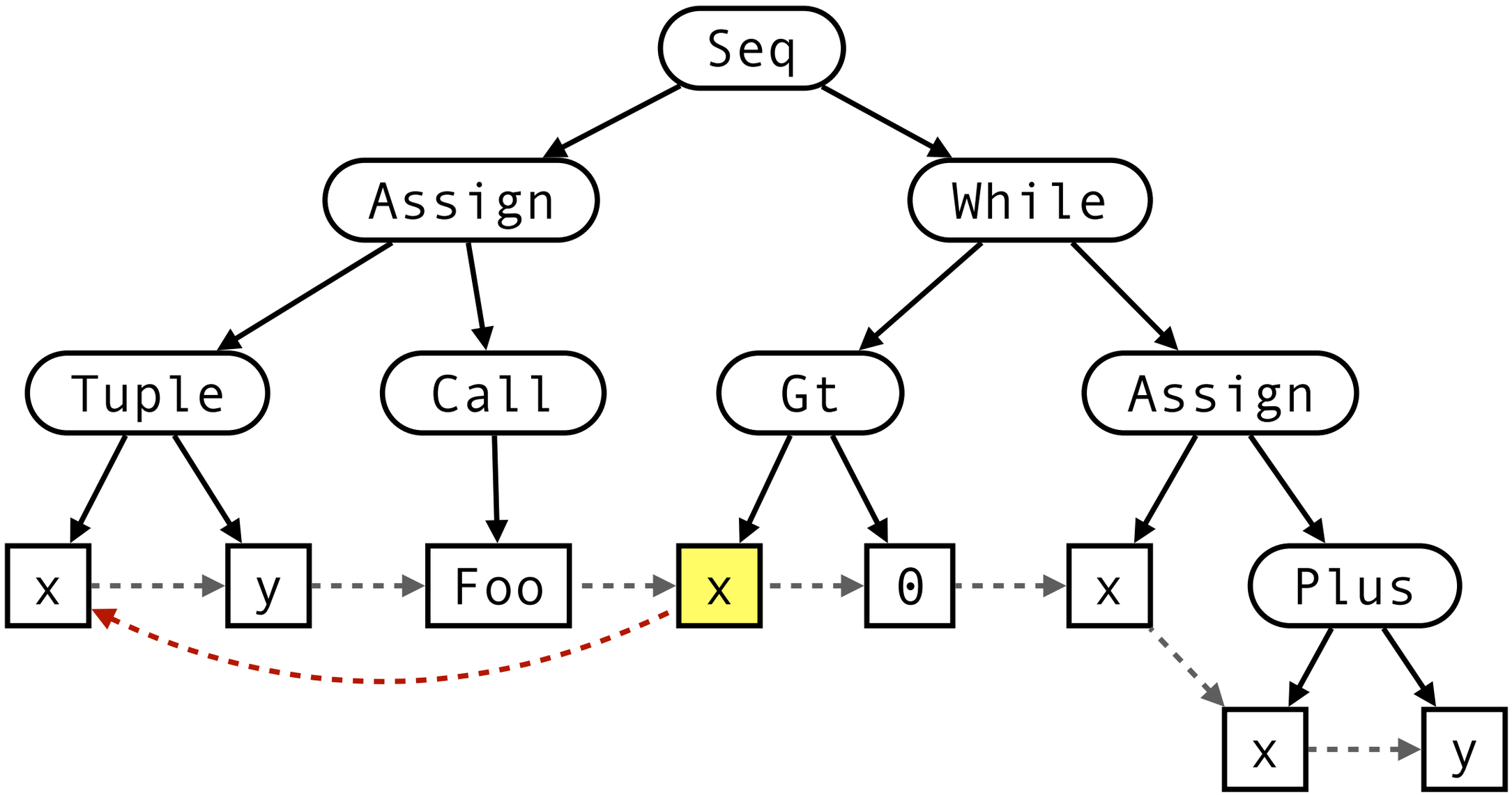
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



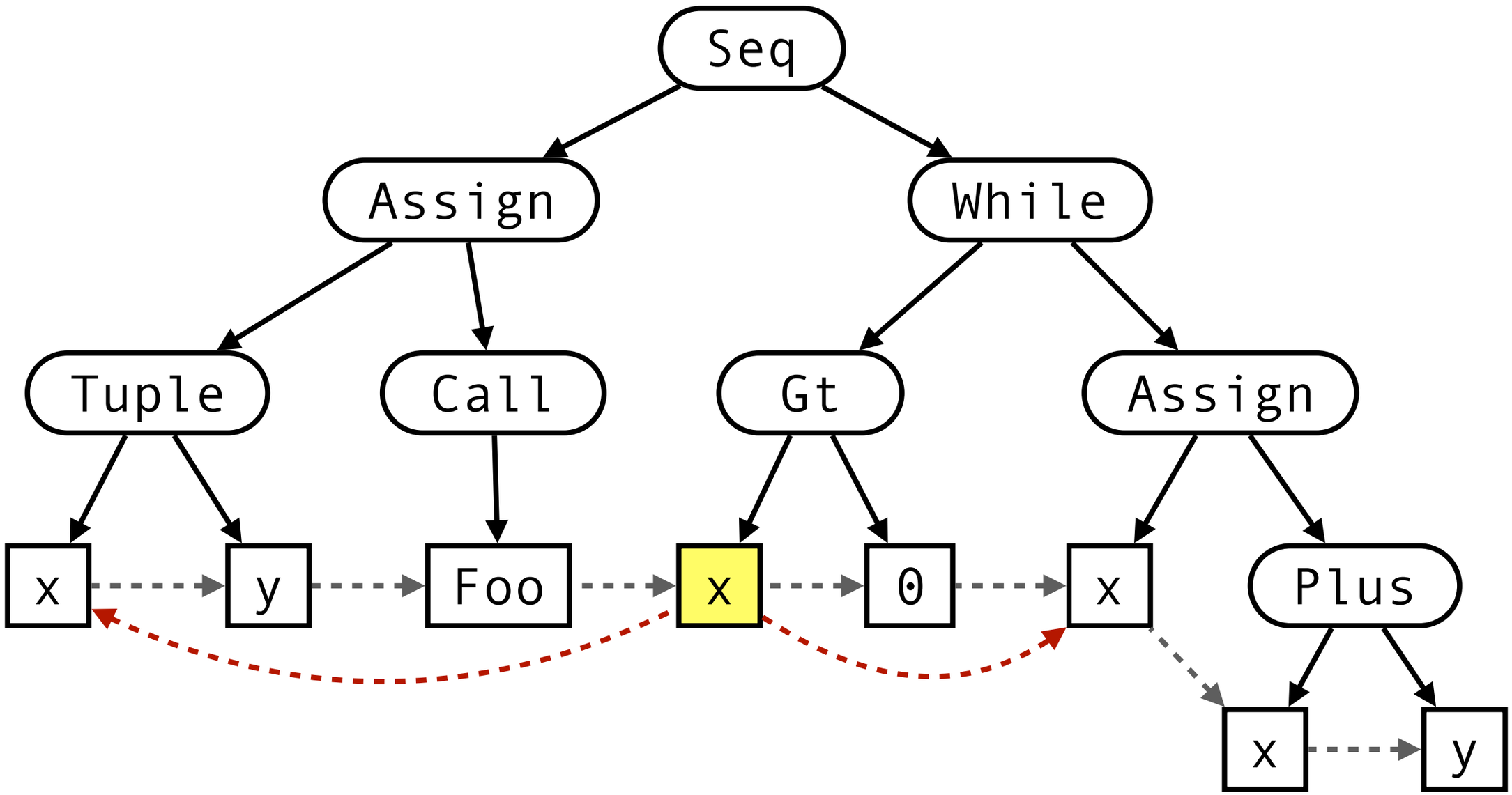
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



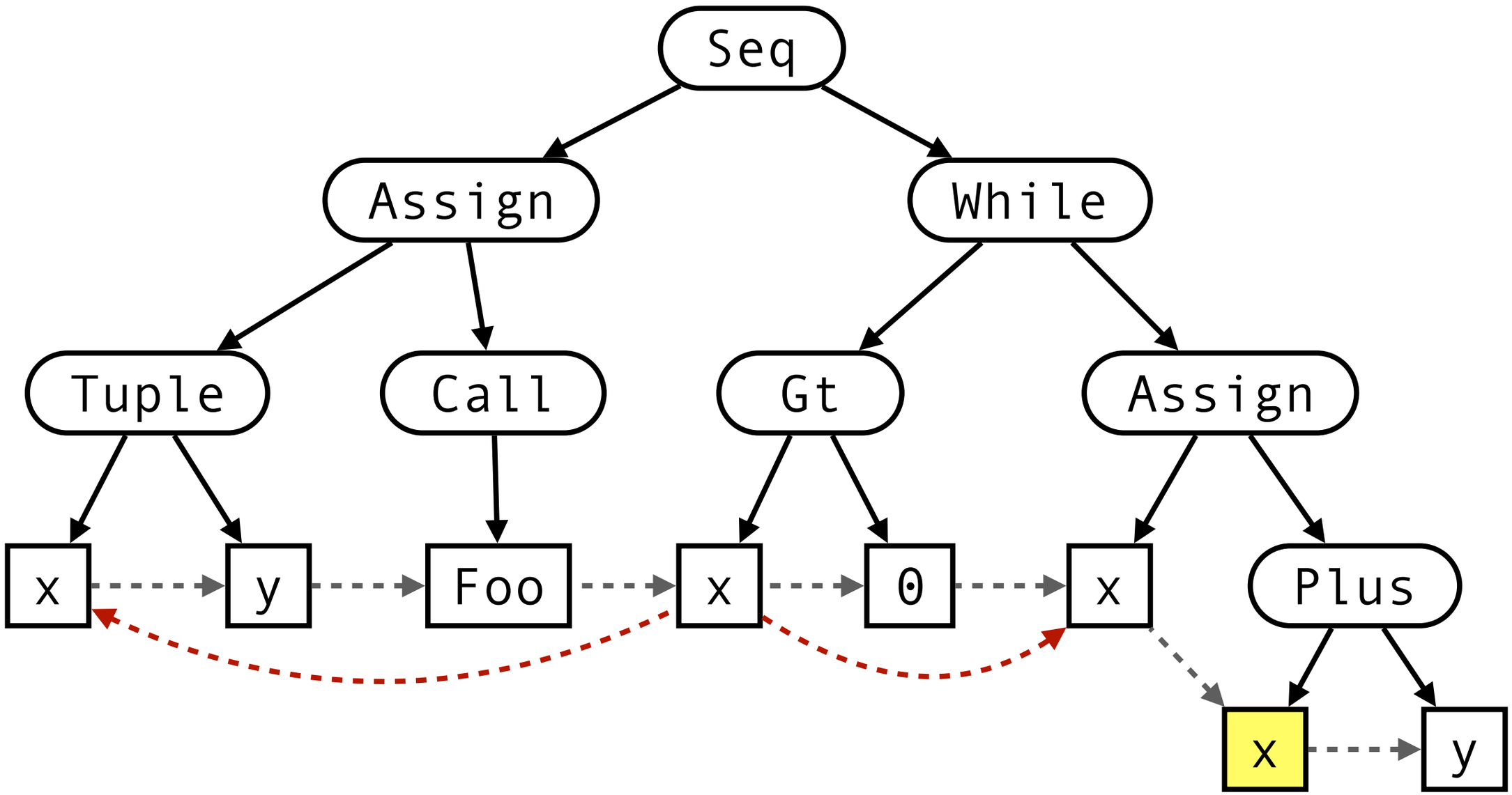
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



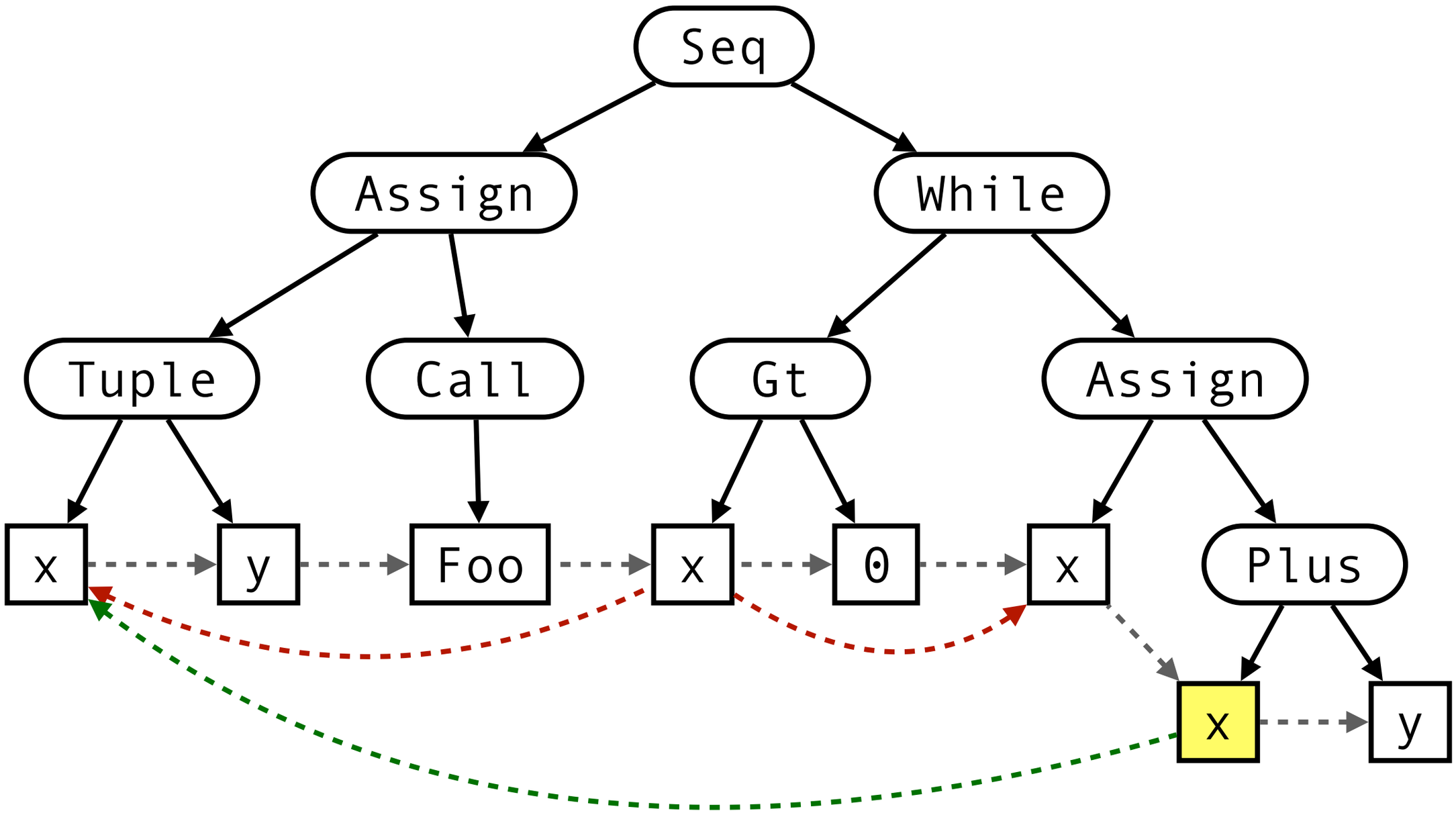
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



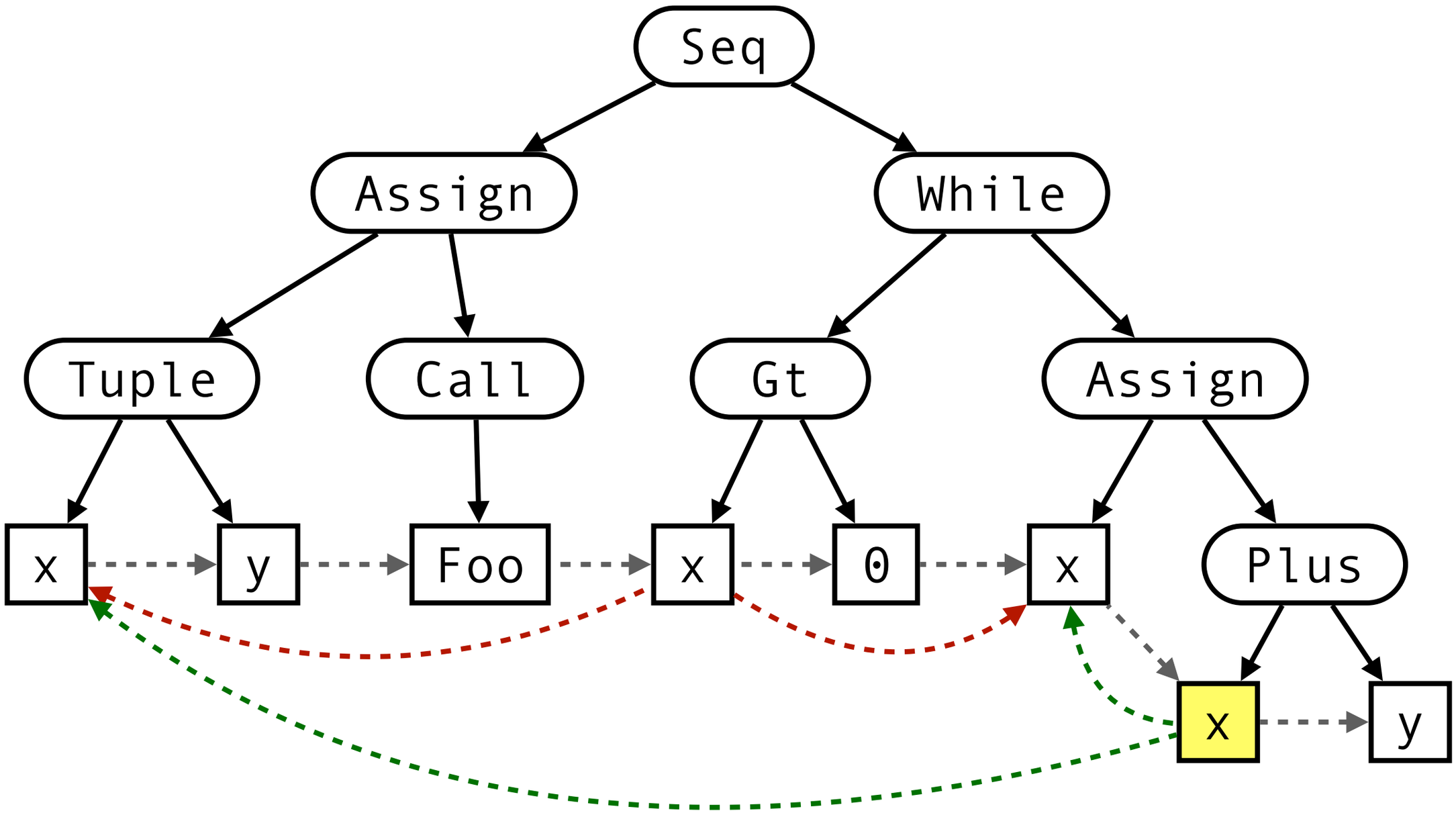
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



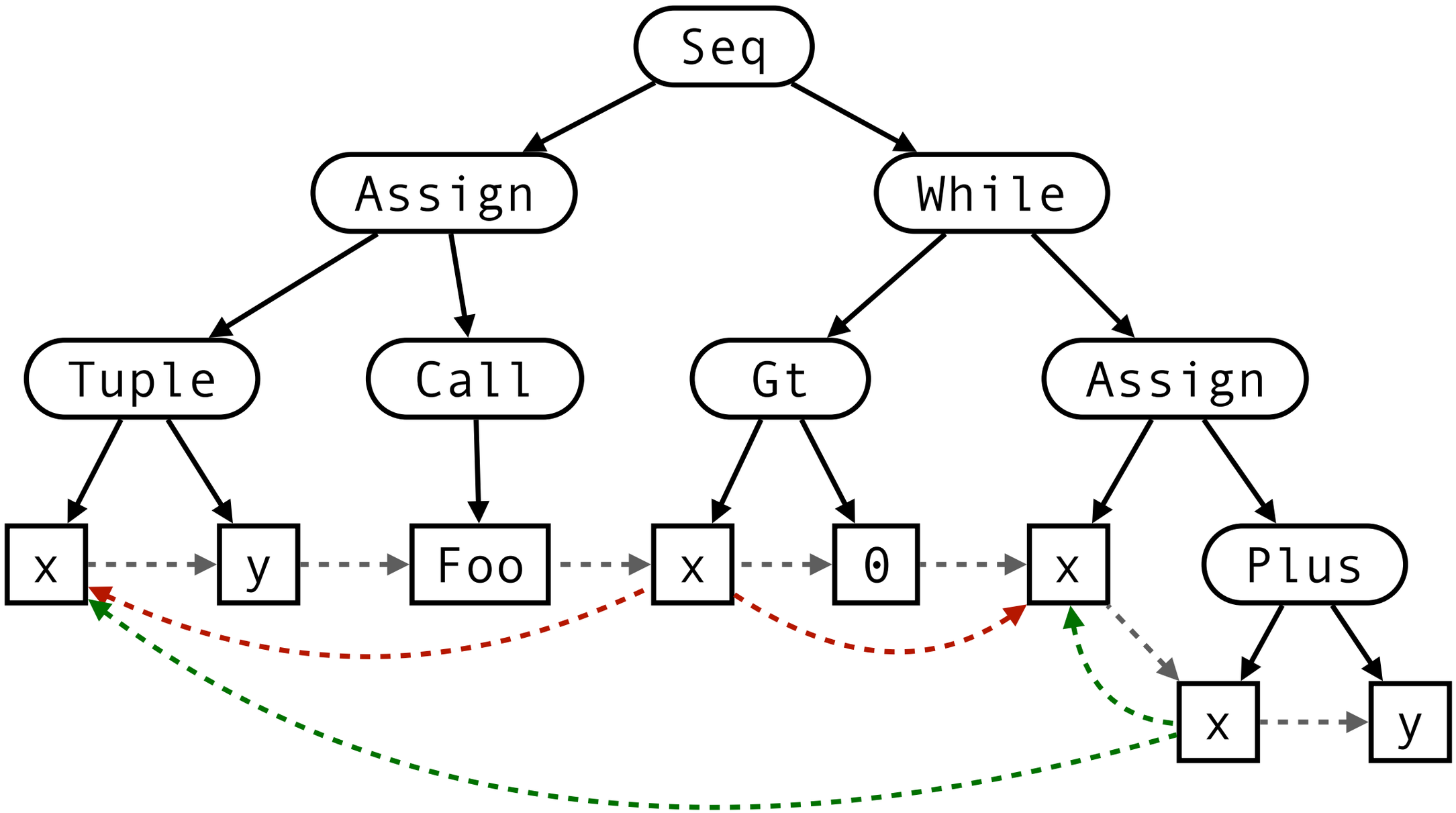
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



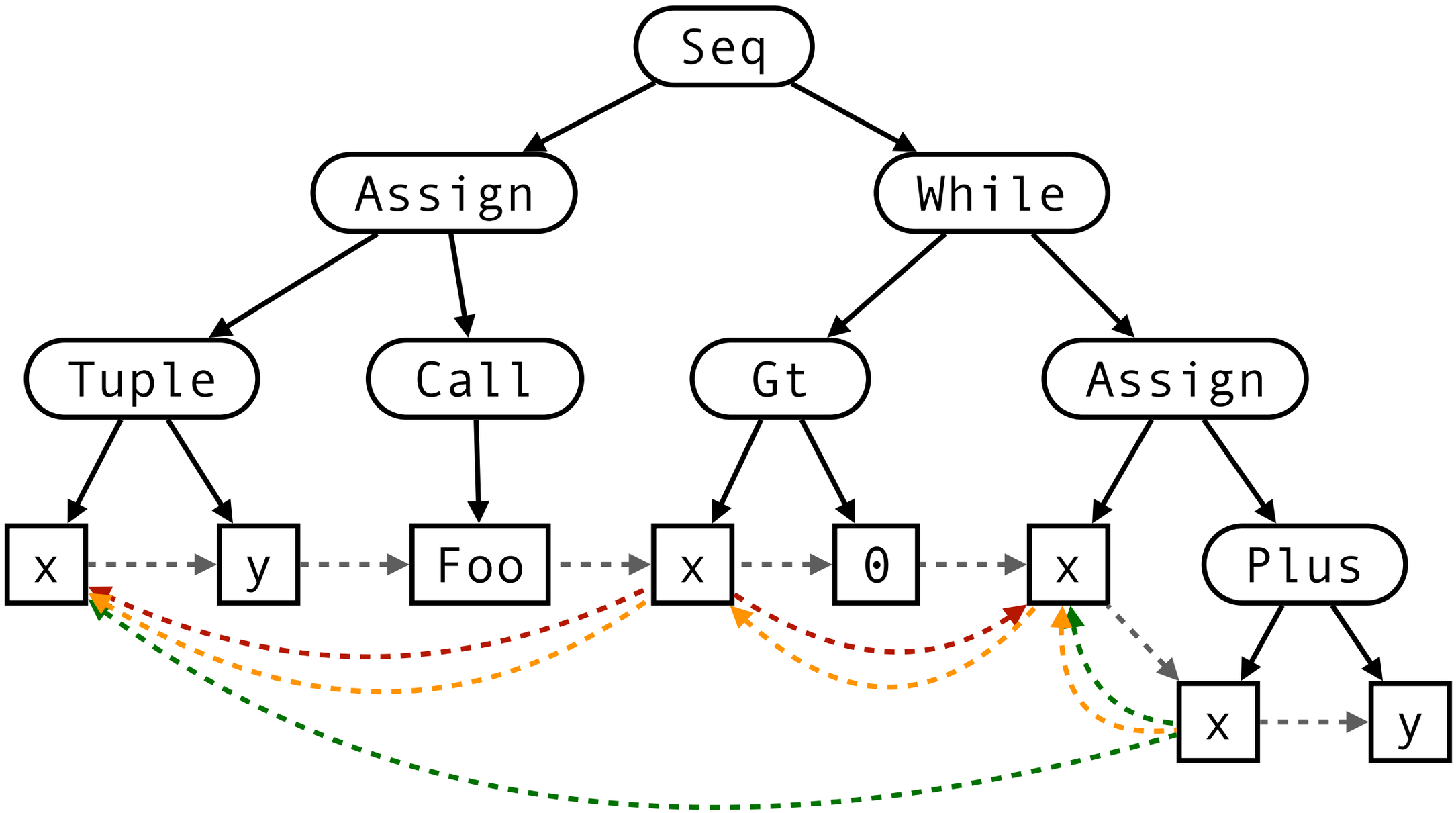
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



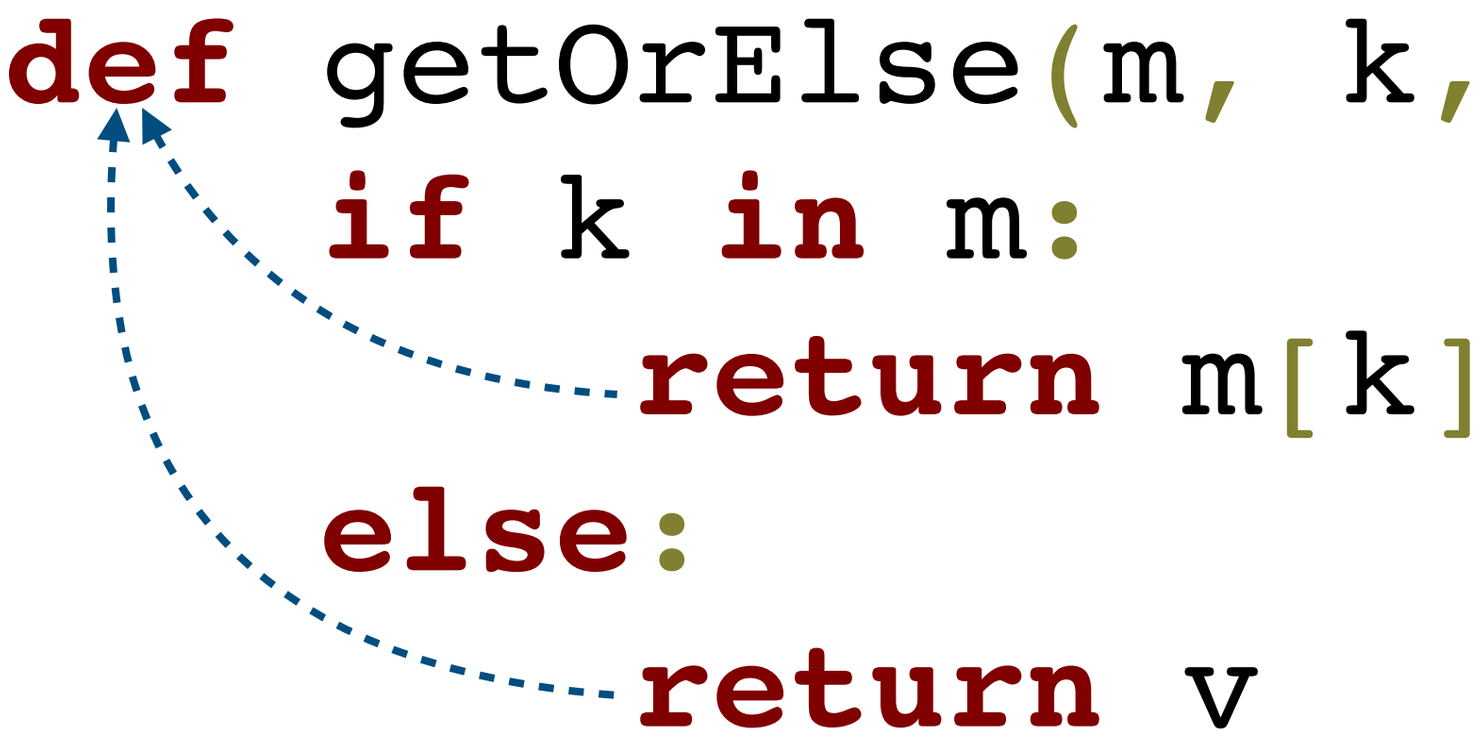
```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



```
(x, y) = Foo()
while (x > 0):
    x = x + y
```



```
def getOrElse(m, k, v):
    if k in m:
        return m[k]
    else:
        return v
```



```
def getOrElse(m, k, v):
    if k in m:
        return m[k]
    else:
        return v
```

```
getOrElse(myMap, someKey, 4)
```

```
def getOrElse(m, k, v):
    if k in m:
        return m[k]
    else:
        return v
```

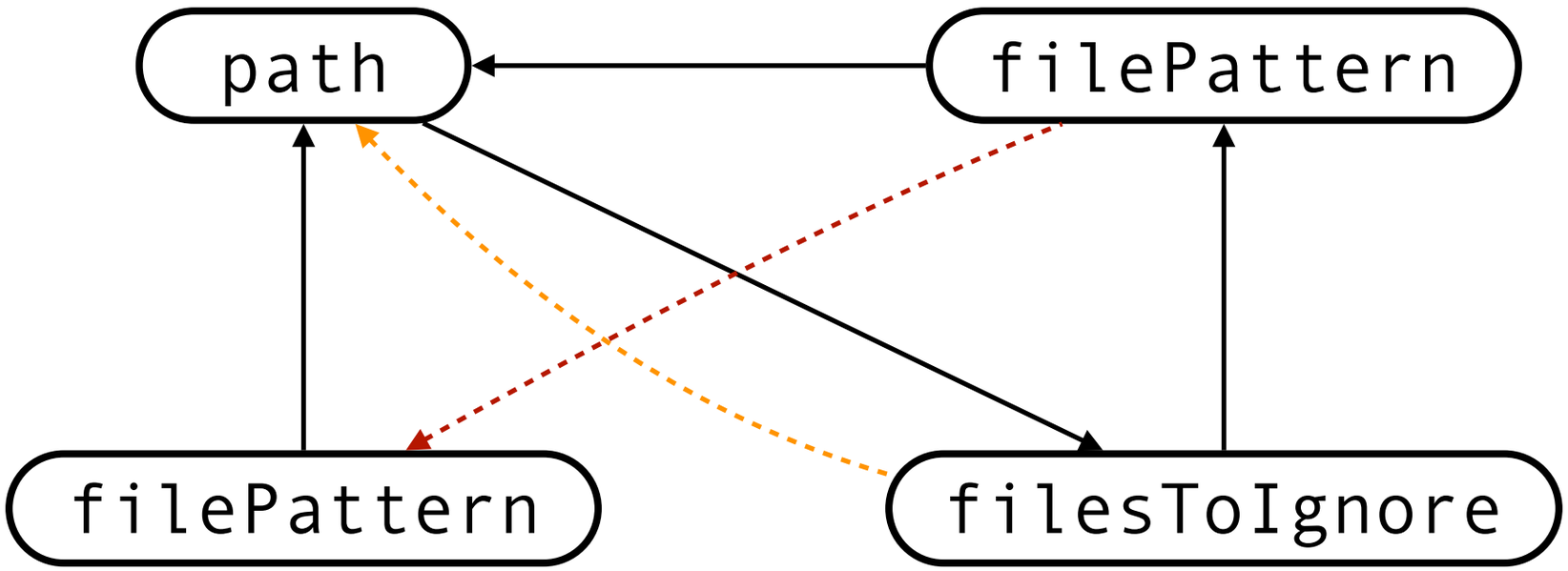
```
getOrElse(myMap, someKey, 4)
```

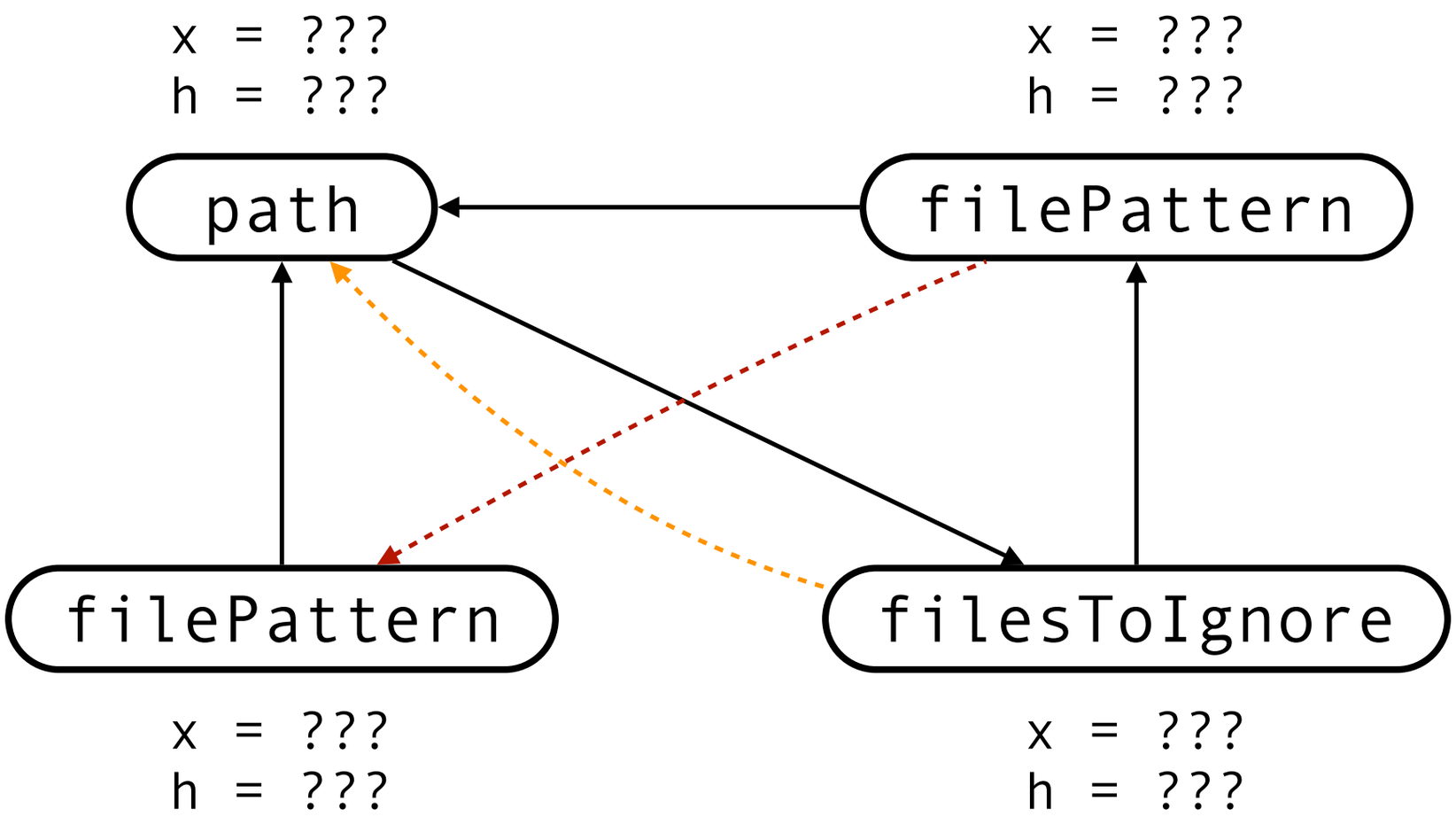
```
def getOrElse(m, k, v):
    if k in m:
        return m[k]
    else:
        return v
```

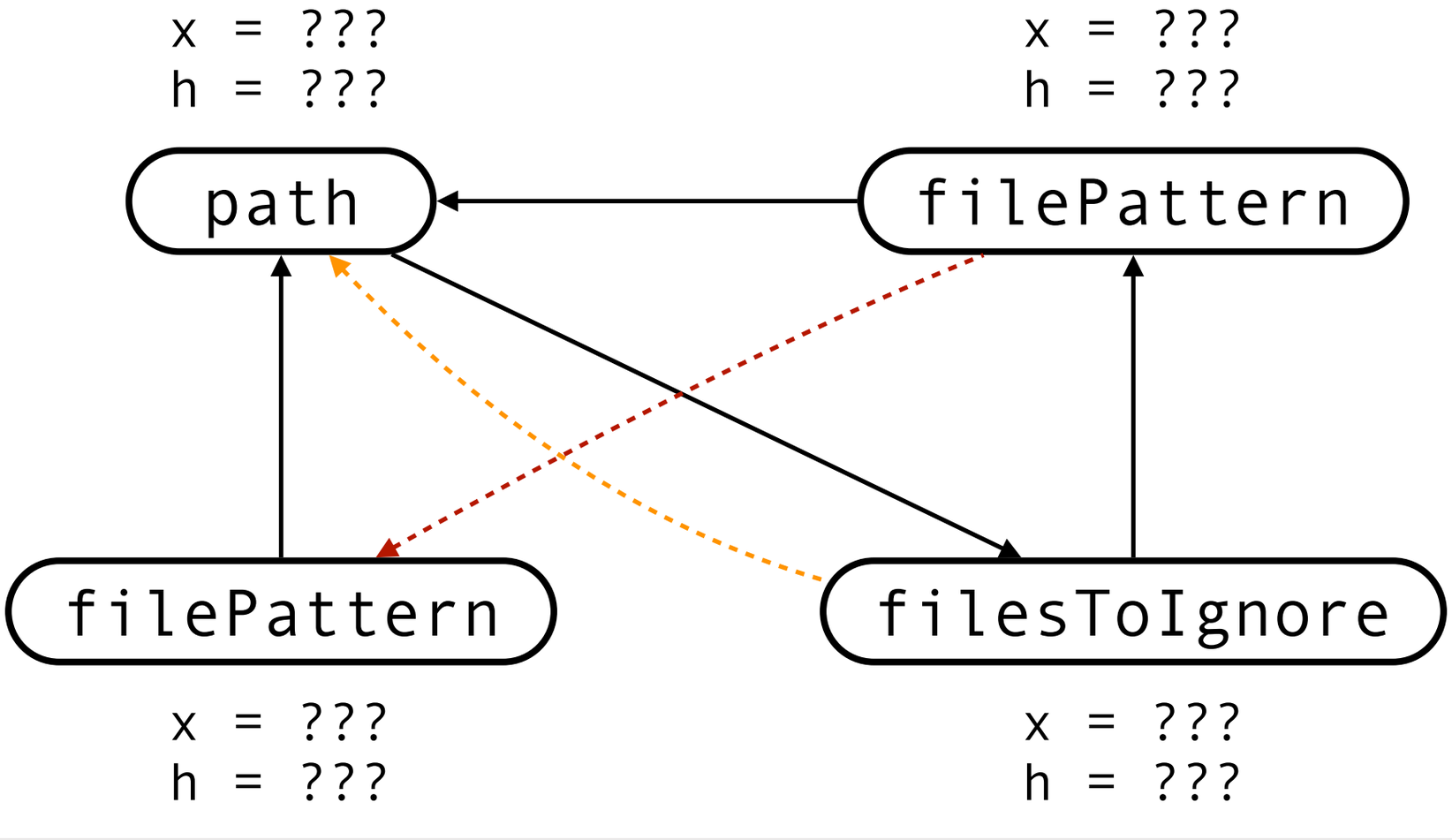
```
def max(x, y):
    if x > y:
        return x
    else:
        return y
```

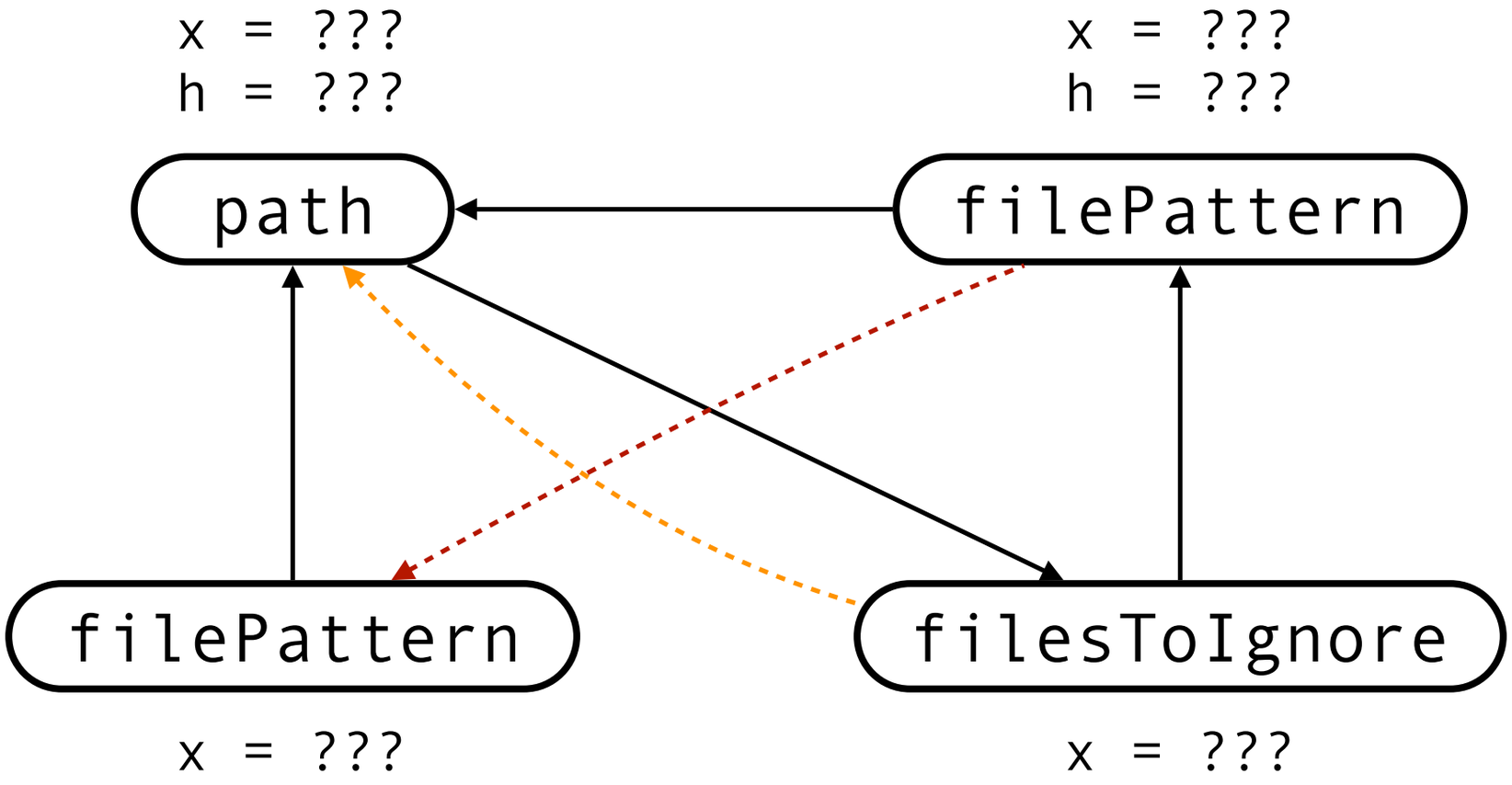
```
def max(x, y):
    if x > y:
        return x
    else:
        return y
```

```
def max(x, y):
    if x > y:
        return x
    else:
        return y
```

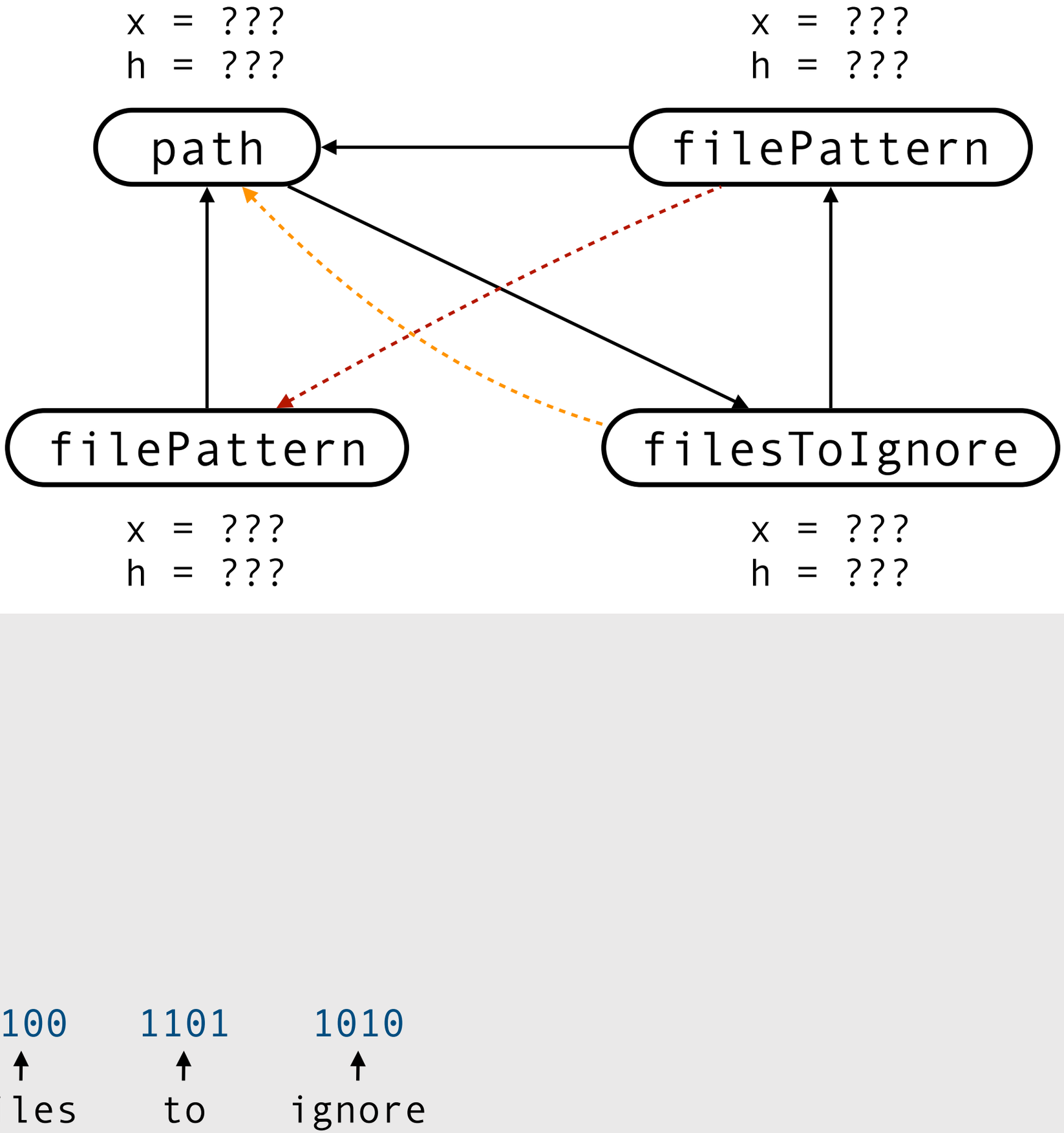

path
filePattern
filePattern
filesToIgnore


x = ???
h = ???
x = ???
h = ???
path
filePattern
filePattern
filesToIgnore
x = ???
h = ???
x = ???
h = ???

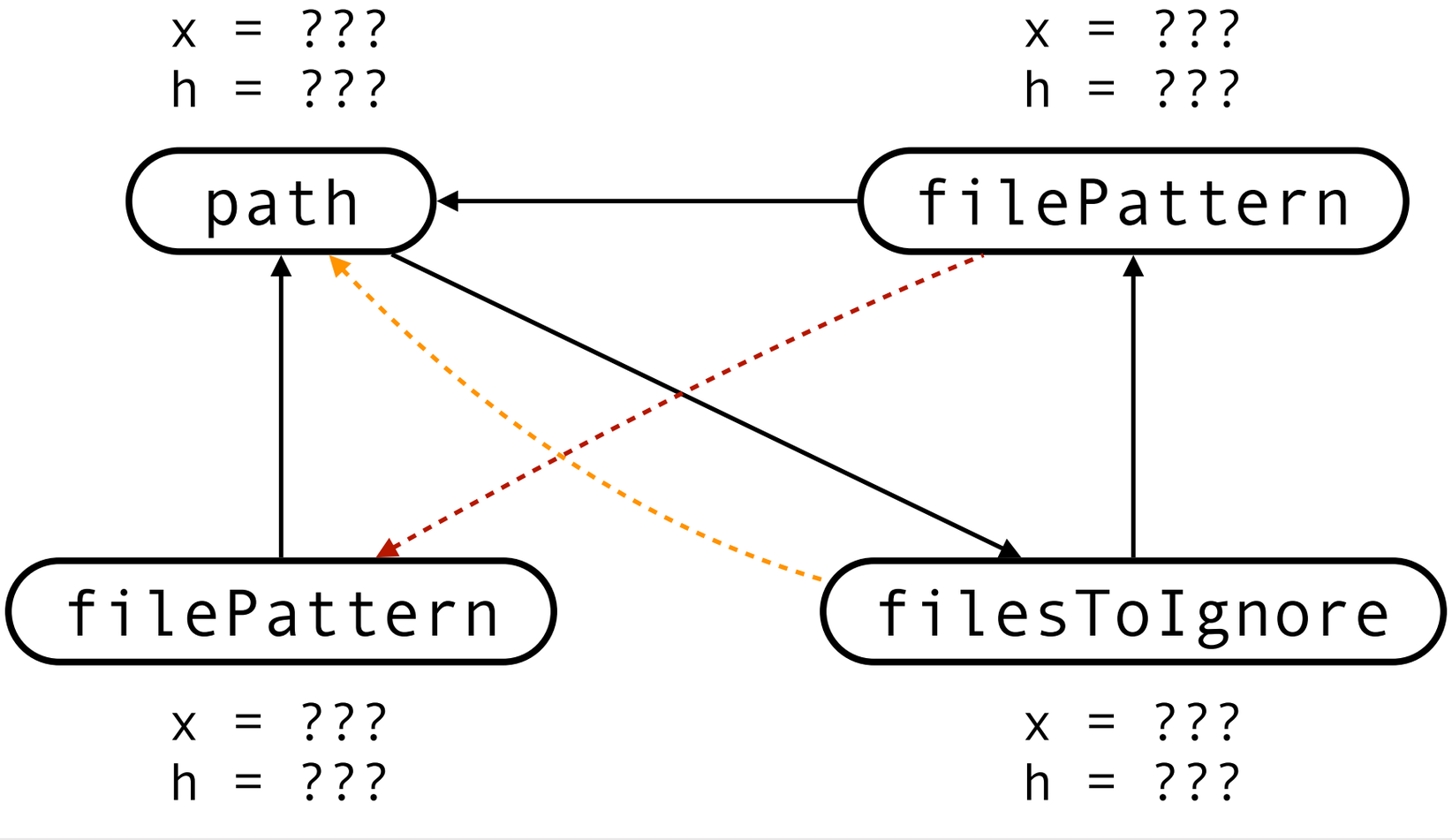

x = ???
h = ???
x = ???
h = ???
path
filePattern
filePattern
filesToIgnore
x = ???
h = ???
x = ???
h = ???

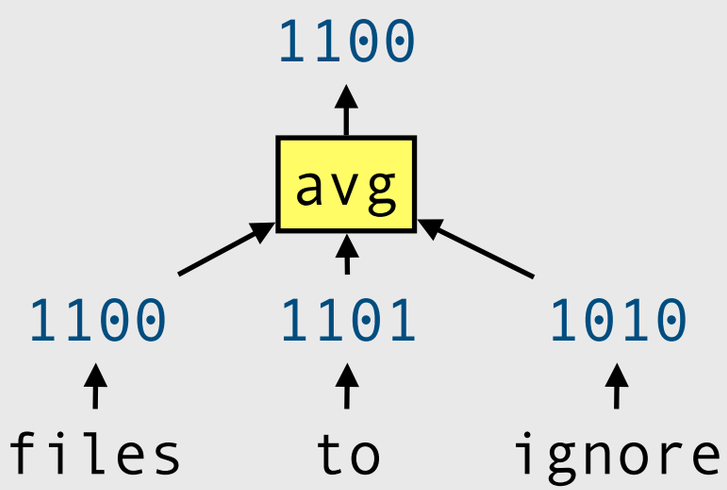


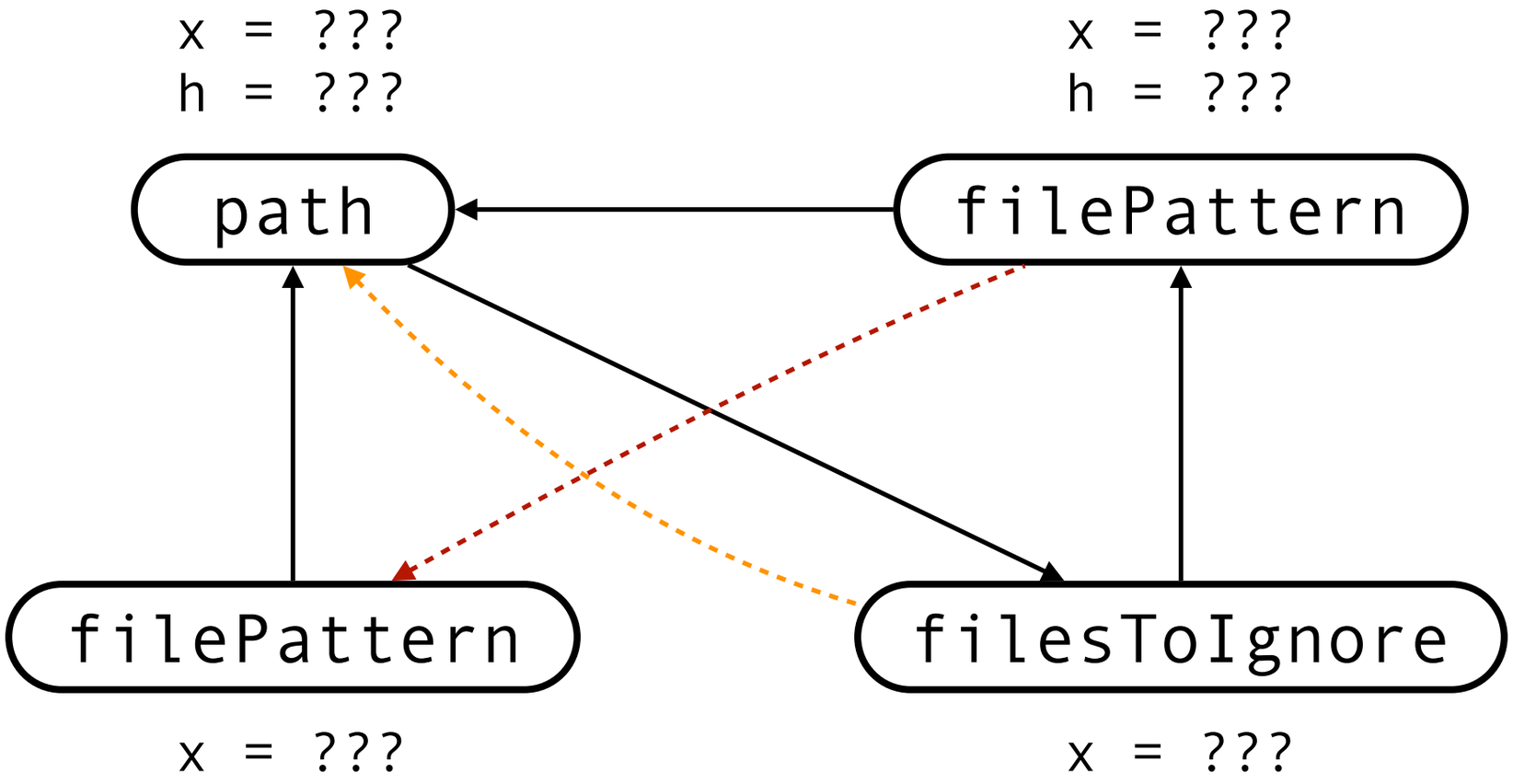
files to ignore

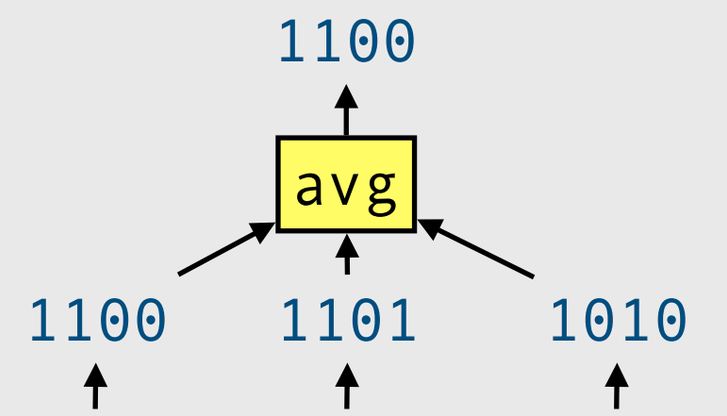



```
1100    1101    1010
 ↑       ↑       ↑
files    to    ignore
```

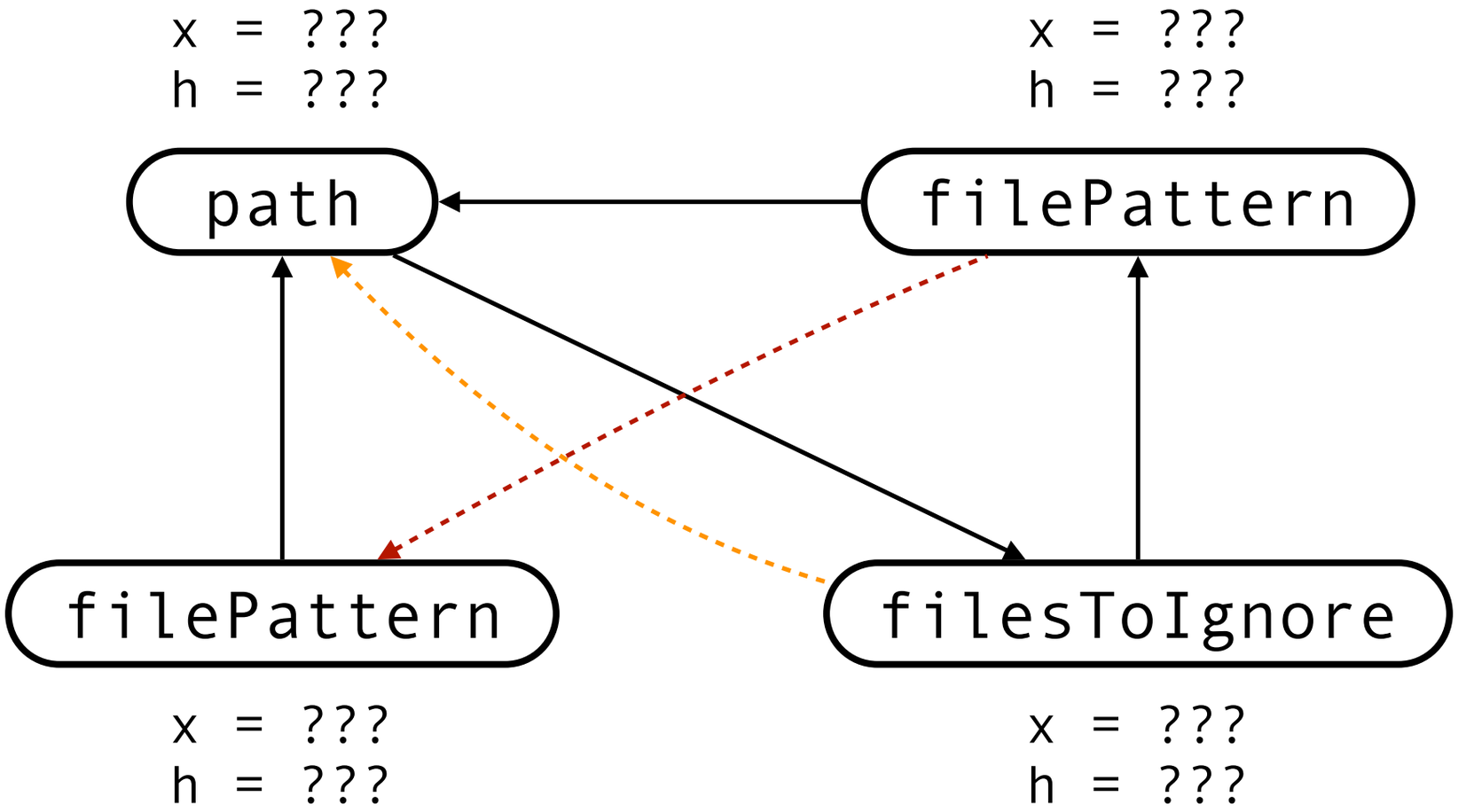

x = ???
h = ???
x = ???
h = ???
path
filePattern
filePattern
filesToIgnore
x = ???
h = ???
x = ???
h = ???


1100
avg
1100
1101
1010
files
to
ignore

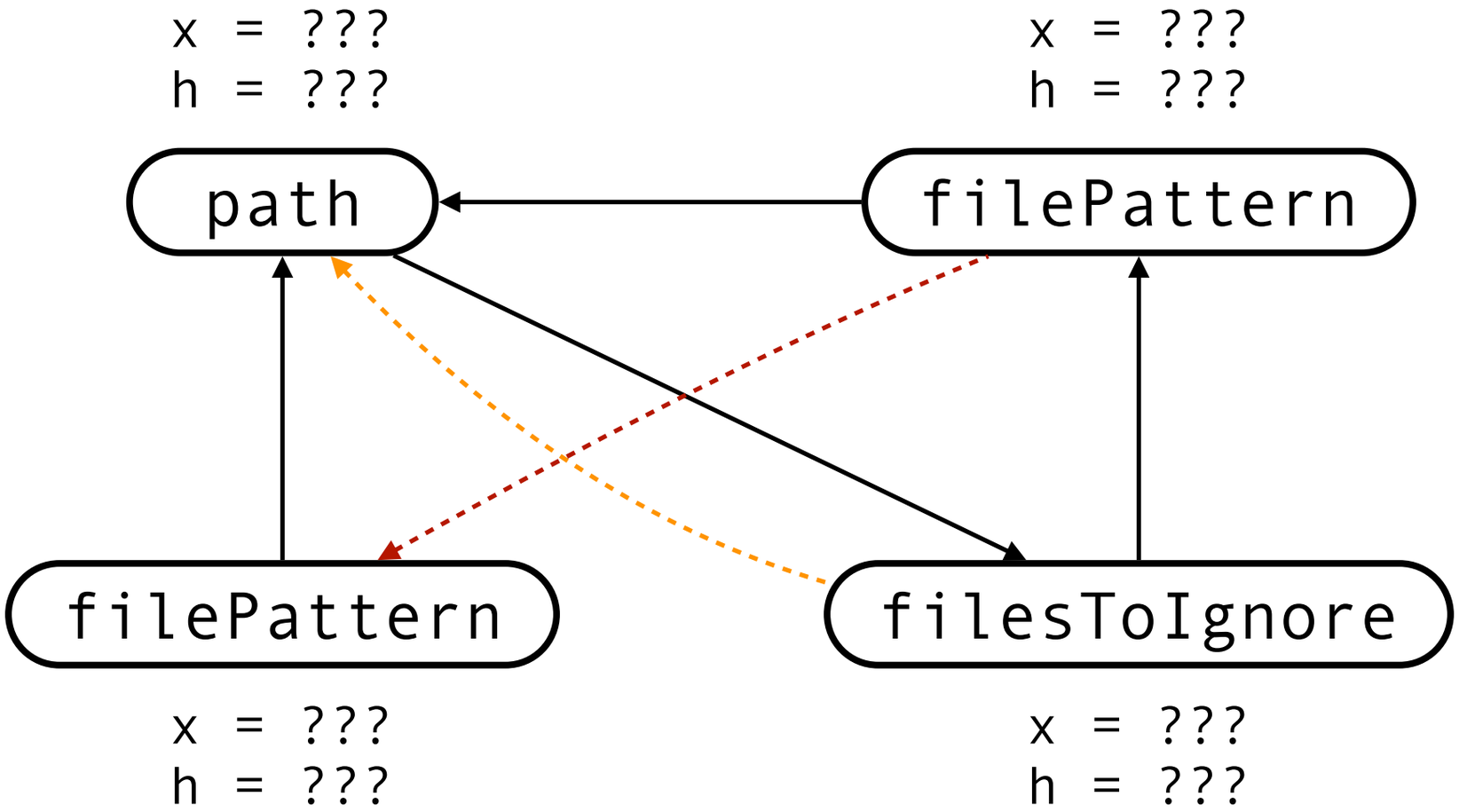


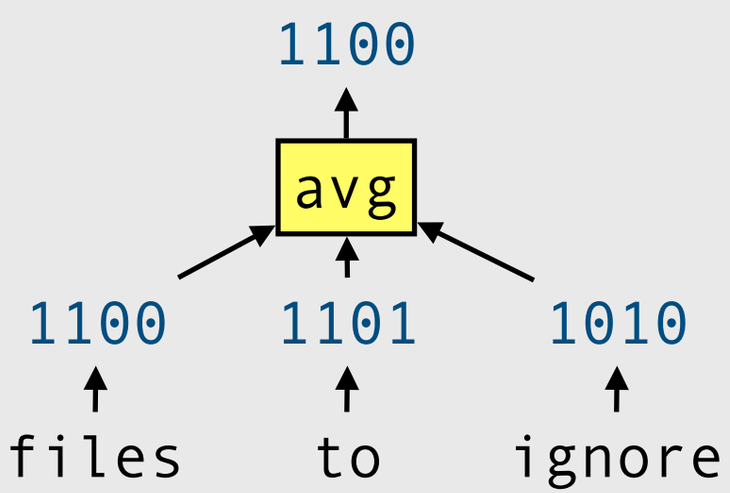


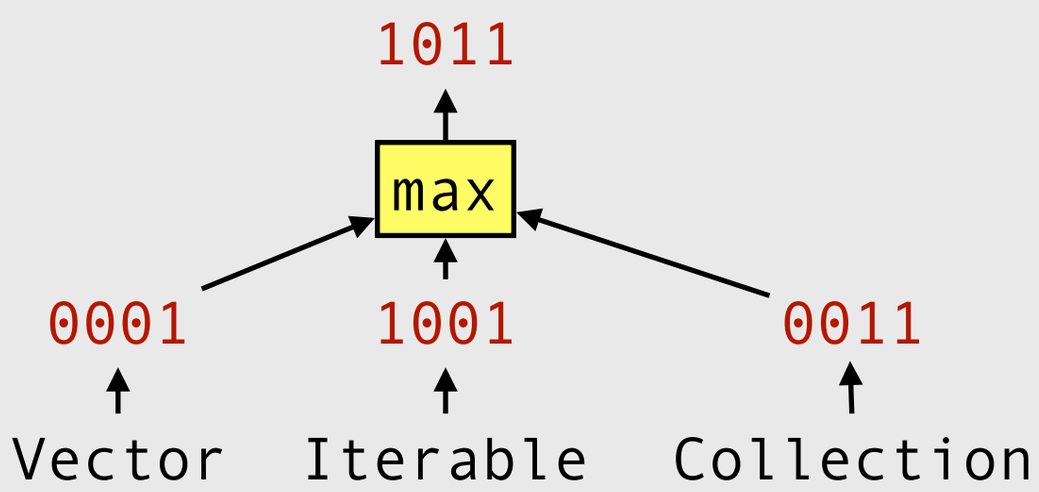
Vector Iterable Collection

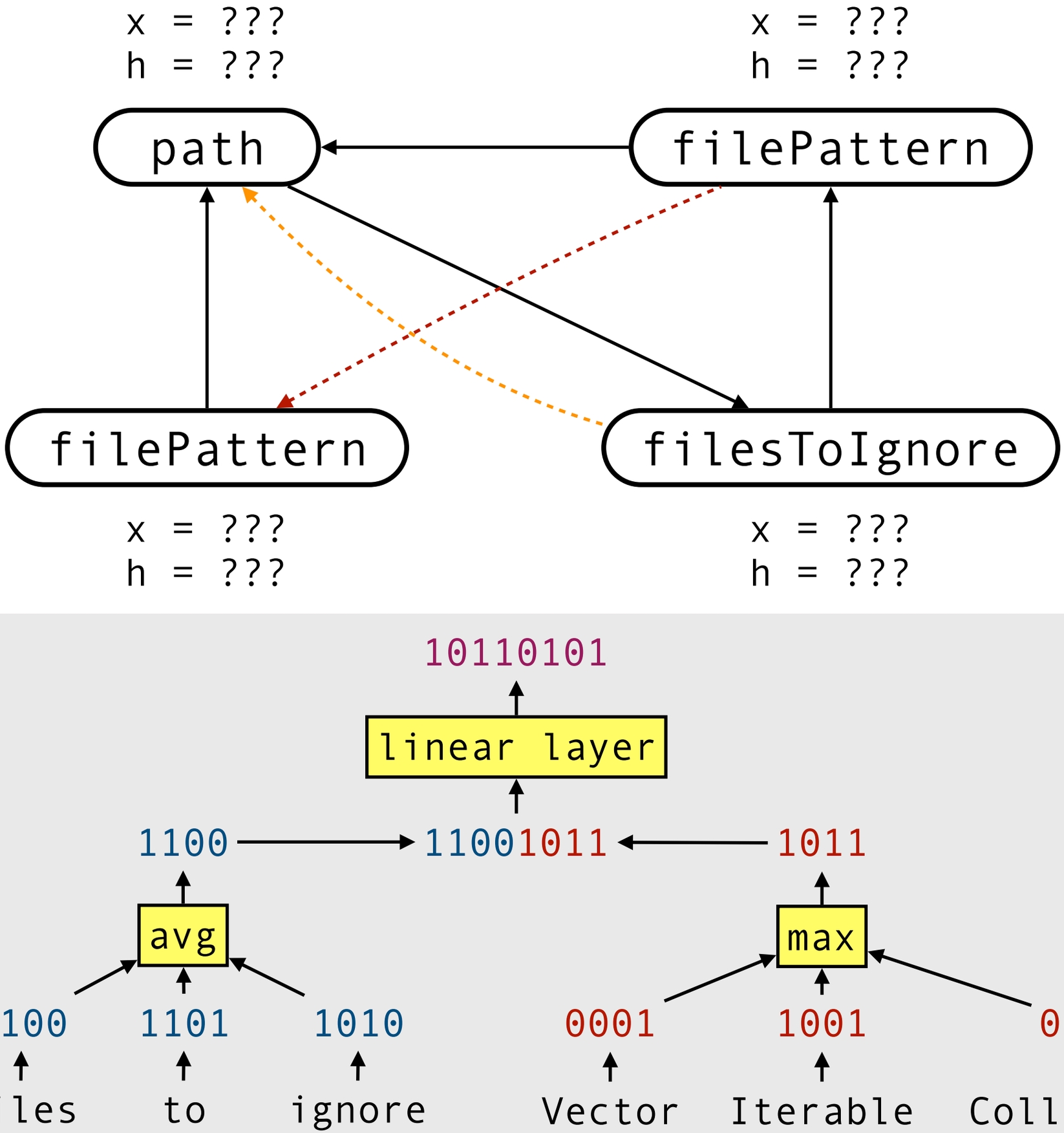

x = ???
h = ???
x = ???
h = ???
path
filePattern
filePattern
filesToIgnore
x = ???
h = ???
x = ???
h = ???

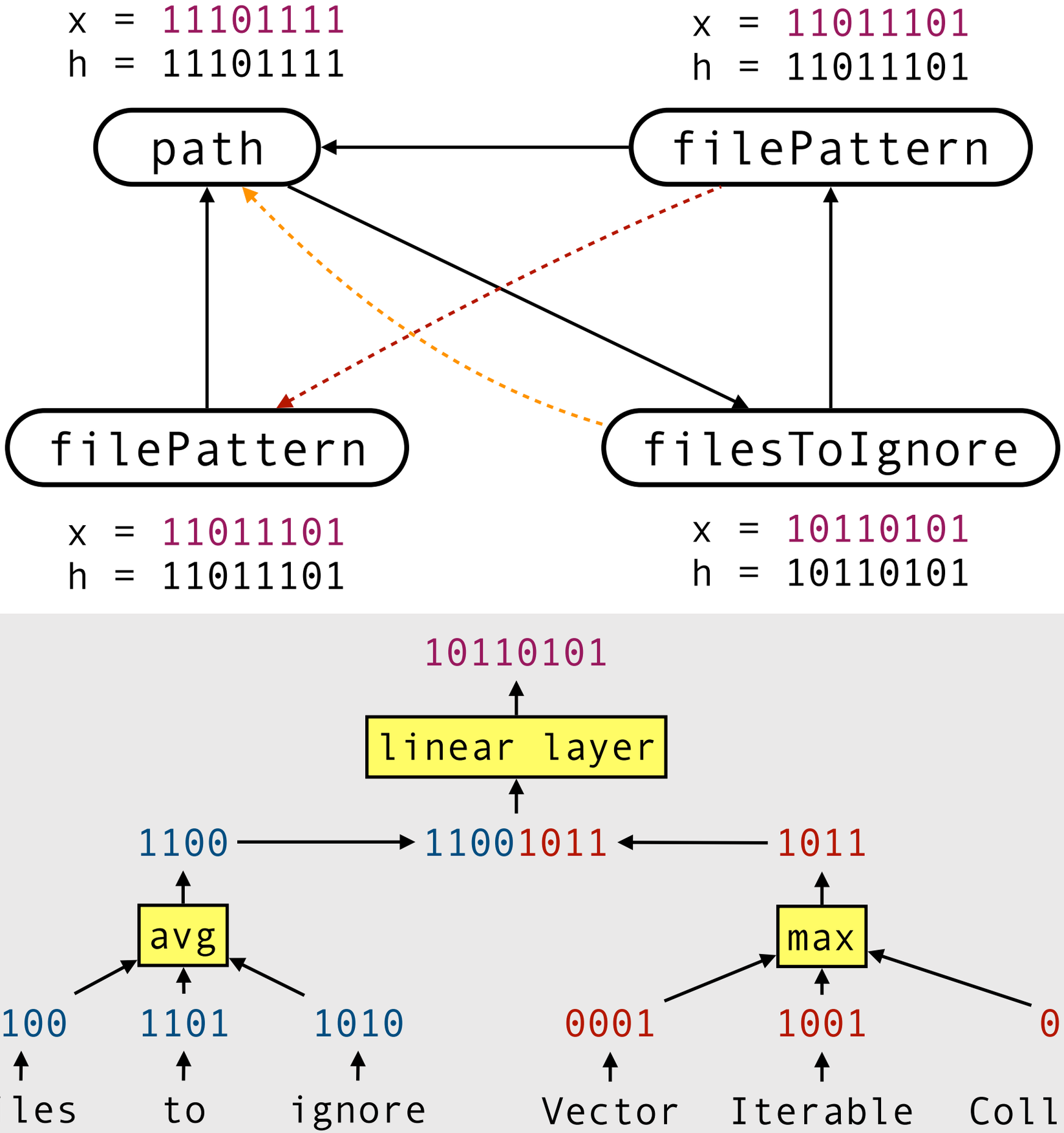

1100
avg
1100
1101
1010
files
to
ignore
0001
1001
0011
Vector
Iterable
Collection

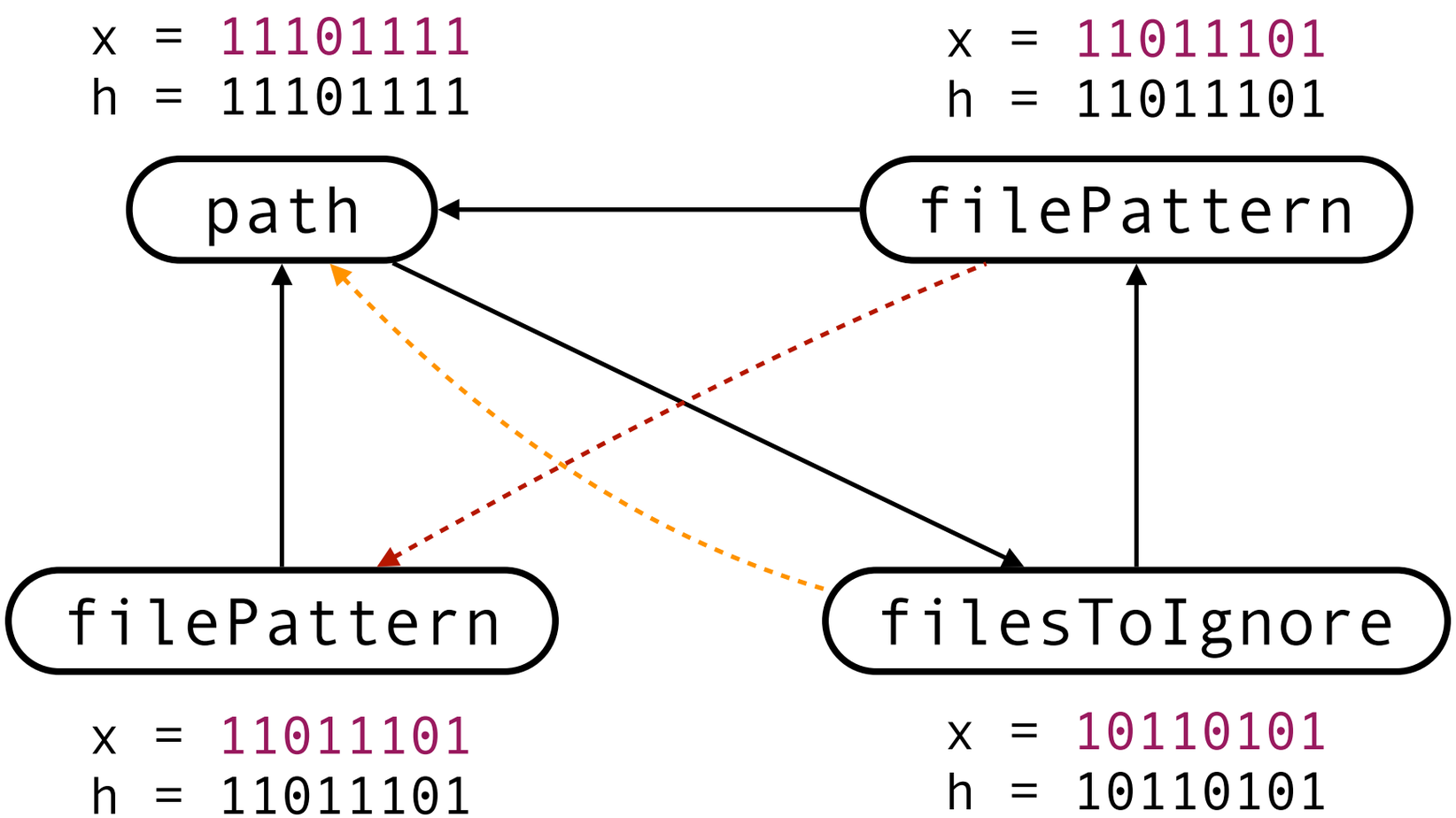

x = ???
h = ???
x = ???
h = ???
path
filePattern
filePattern
filesToIgnore
x = ???
h = ???
x = ???
h = ???

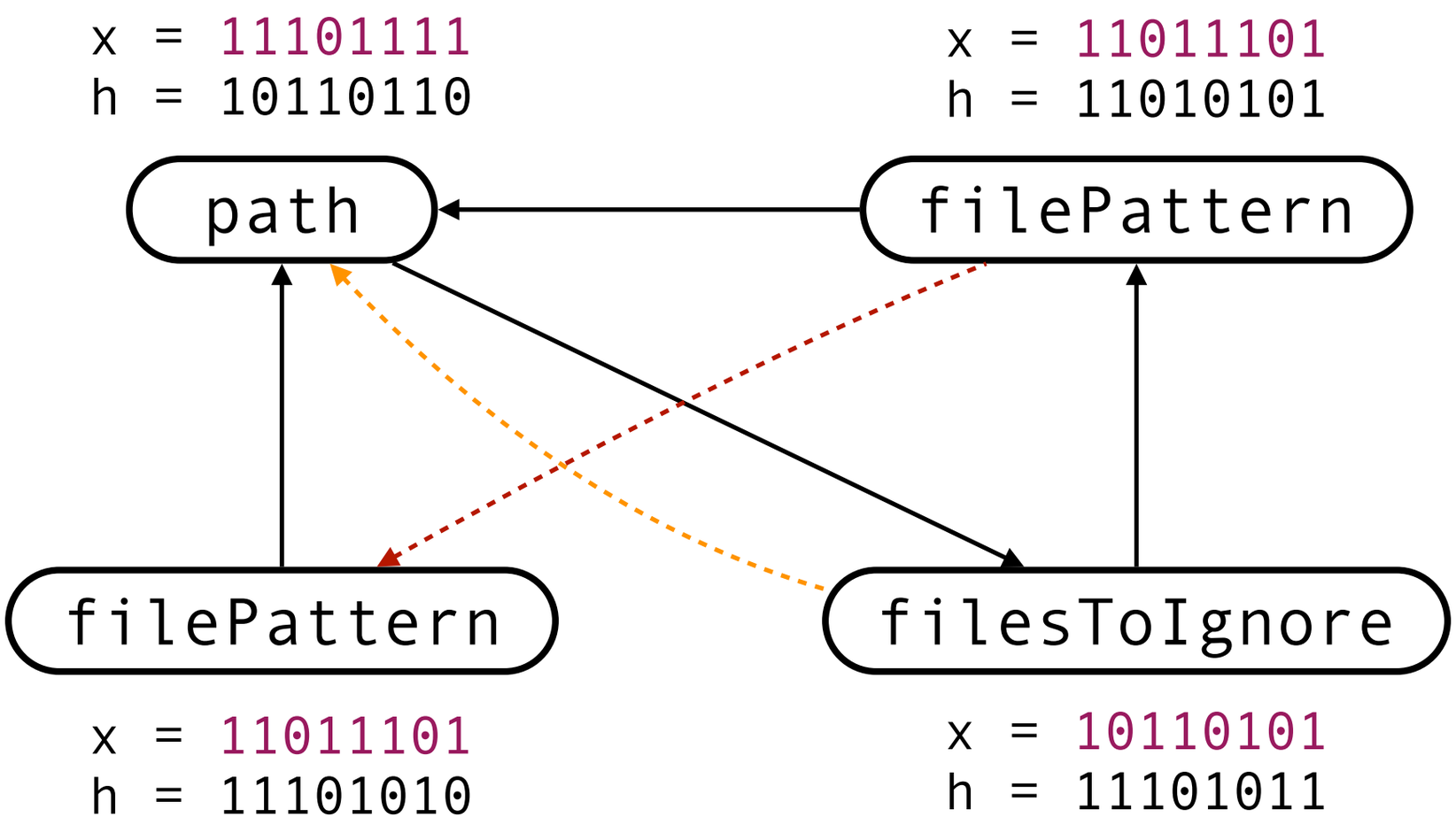

1100
avg
1100
1101
1010
files
to
ignore

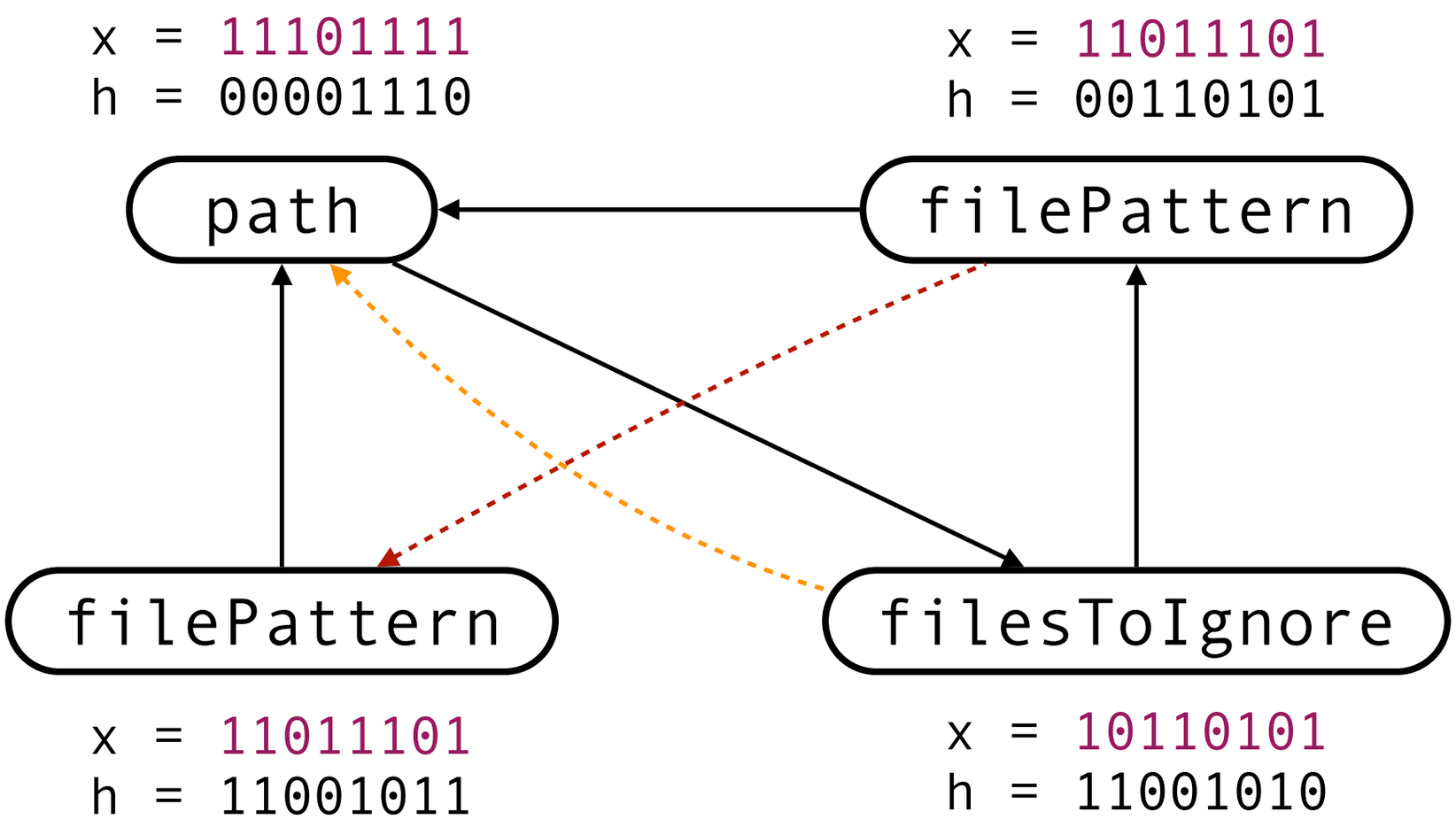

1011
max
0001
1001
0011
Vector
Iterable
Collection

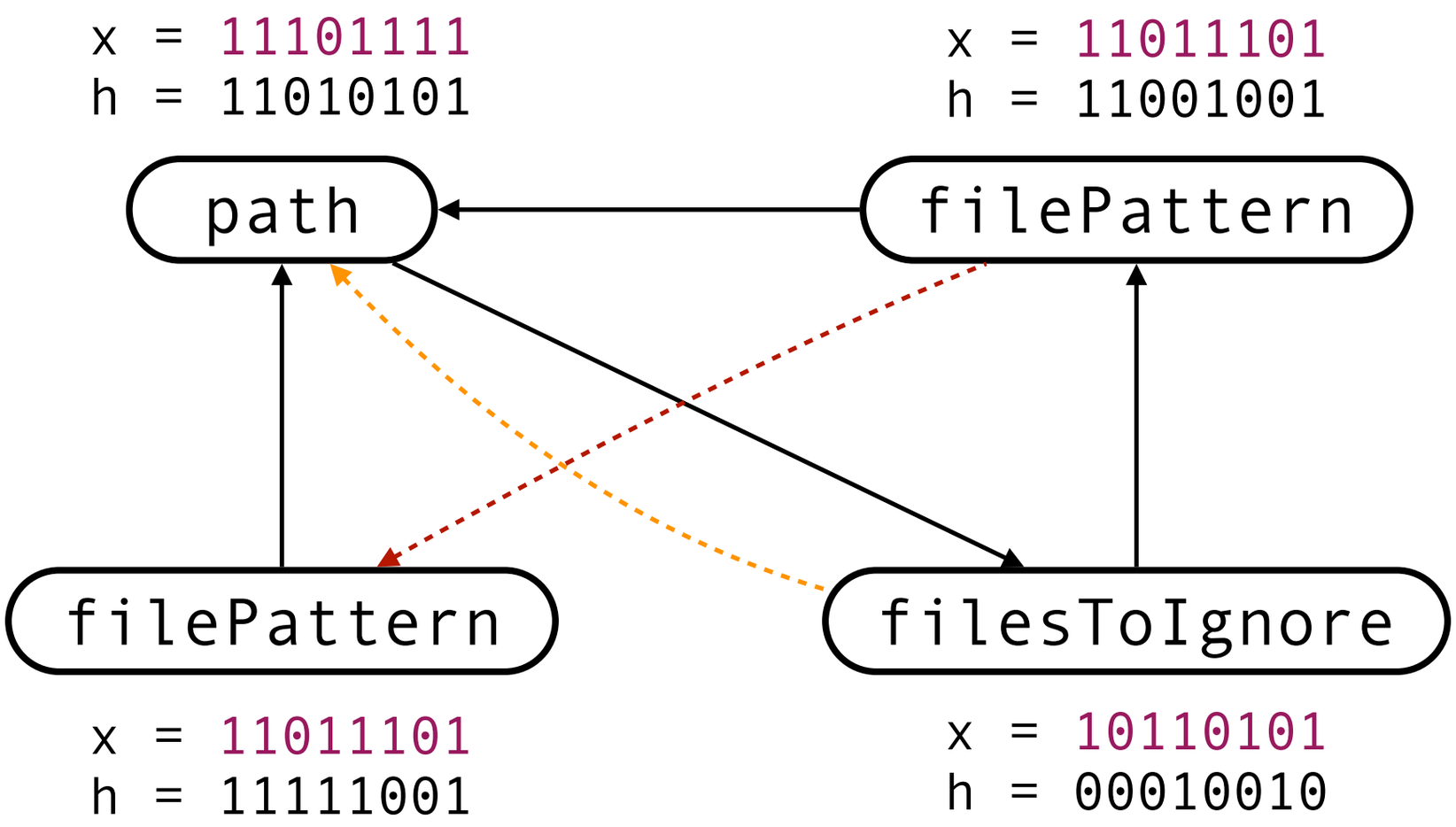

x = ???
h = ???
x = ???
h = ???
path
filePattern
filePattern
filesToIgnore
x = ???
h = ???
x = ???
h = ???
10110101
linear layer
1100
11001011
1011
avg
max
1100
1101
1010
0001
1001
0011
files
to
ignore
Vector
Iterable
Collection

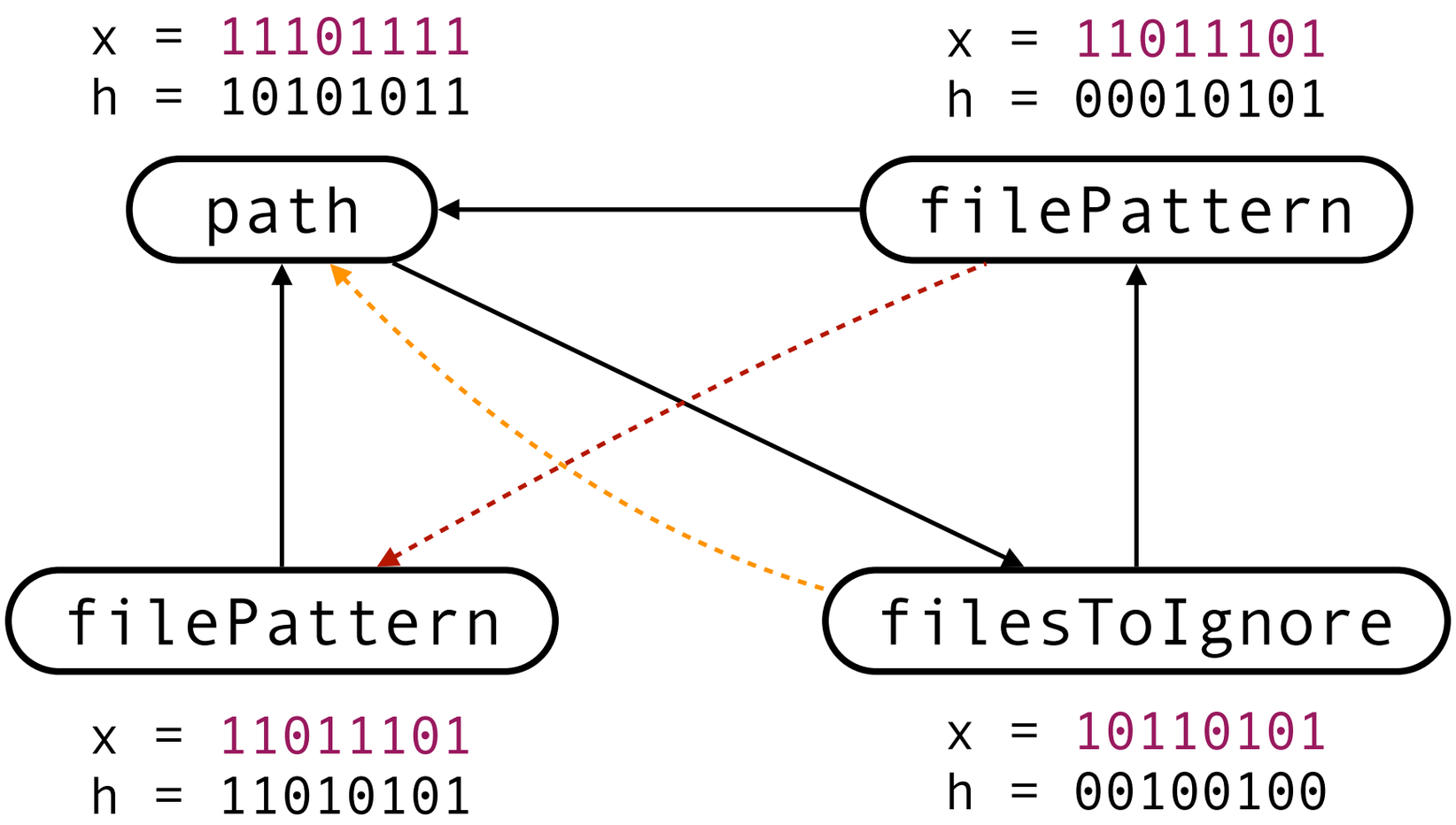

x = 11101111
h = 11101111
x = 11011101
h = 11011101
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11011101
x = 10110101
h = 10110101
10110101
linear layer
1100
11001011
1011
avg
max
1100
1101
1010
0001
1001
0011
files
to
ignore
Vector
Iterable
Collection

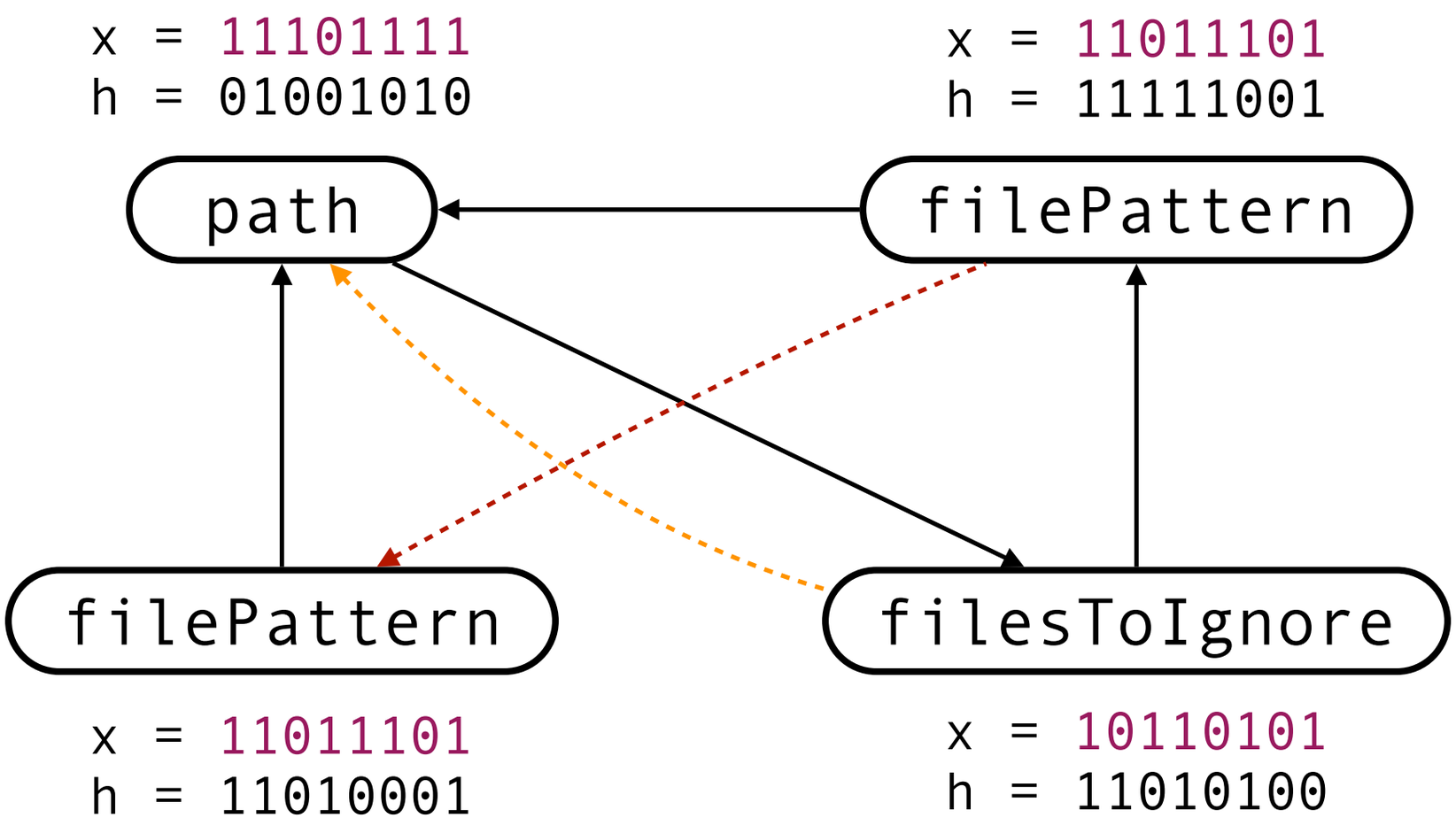

x = 11101111
h = 11101111
x = 11011101
h = 11011101
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11011101
x = 10110101
h = 10110101


x = 11101111
h = 10110110
x = 11011101
h = 11010101
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11101010
x = 10110101
h = 11101011

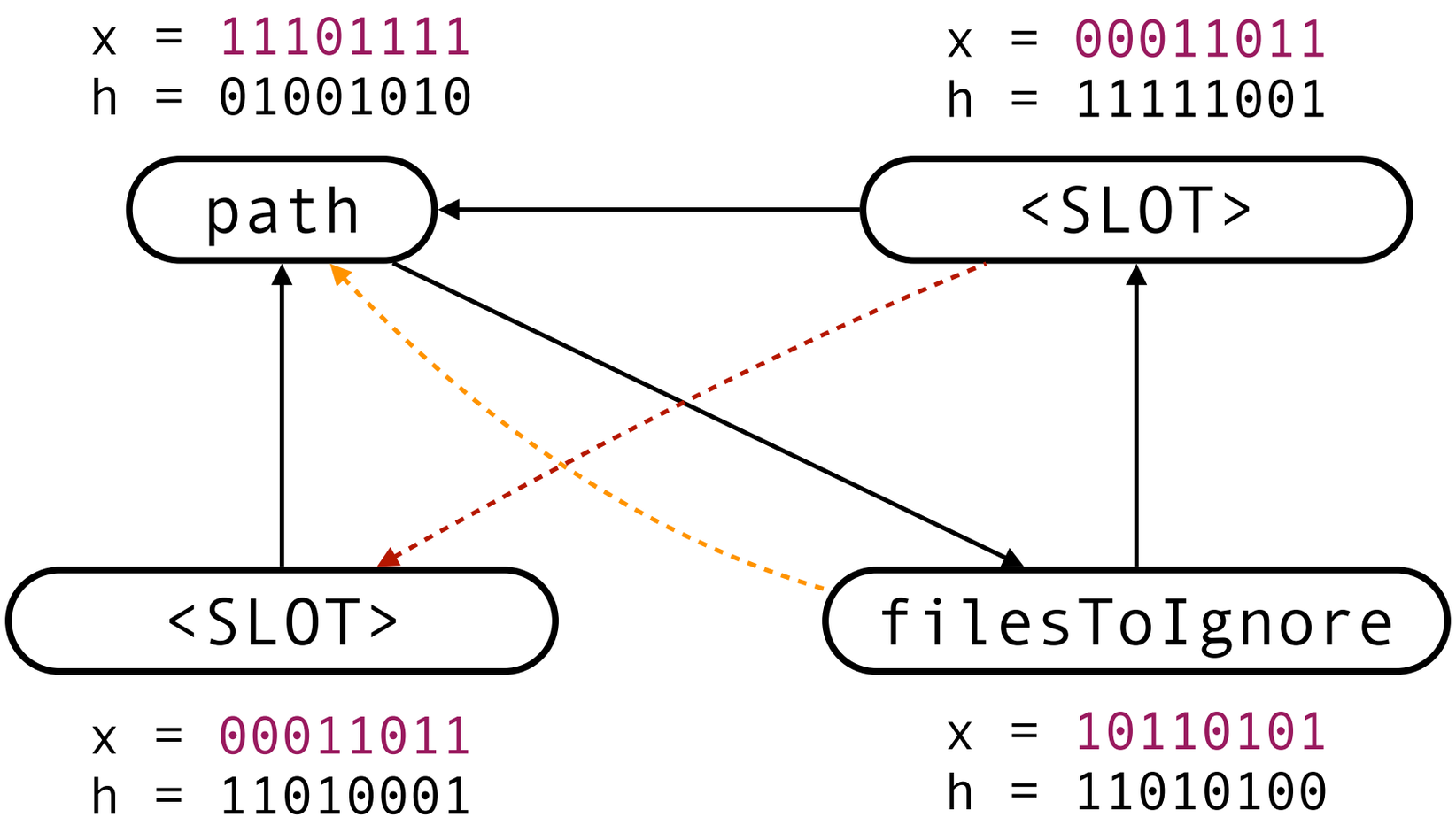

x = 11101111
h = 00001110
x = 11011101
h = 00110101
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11001011
x = 10110101
h = 11001010


x = 11101111
h = 11010101
x = 11011101
h = 11001001
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11111001
x = 10110101
h = 00010010

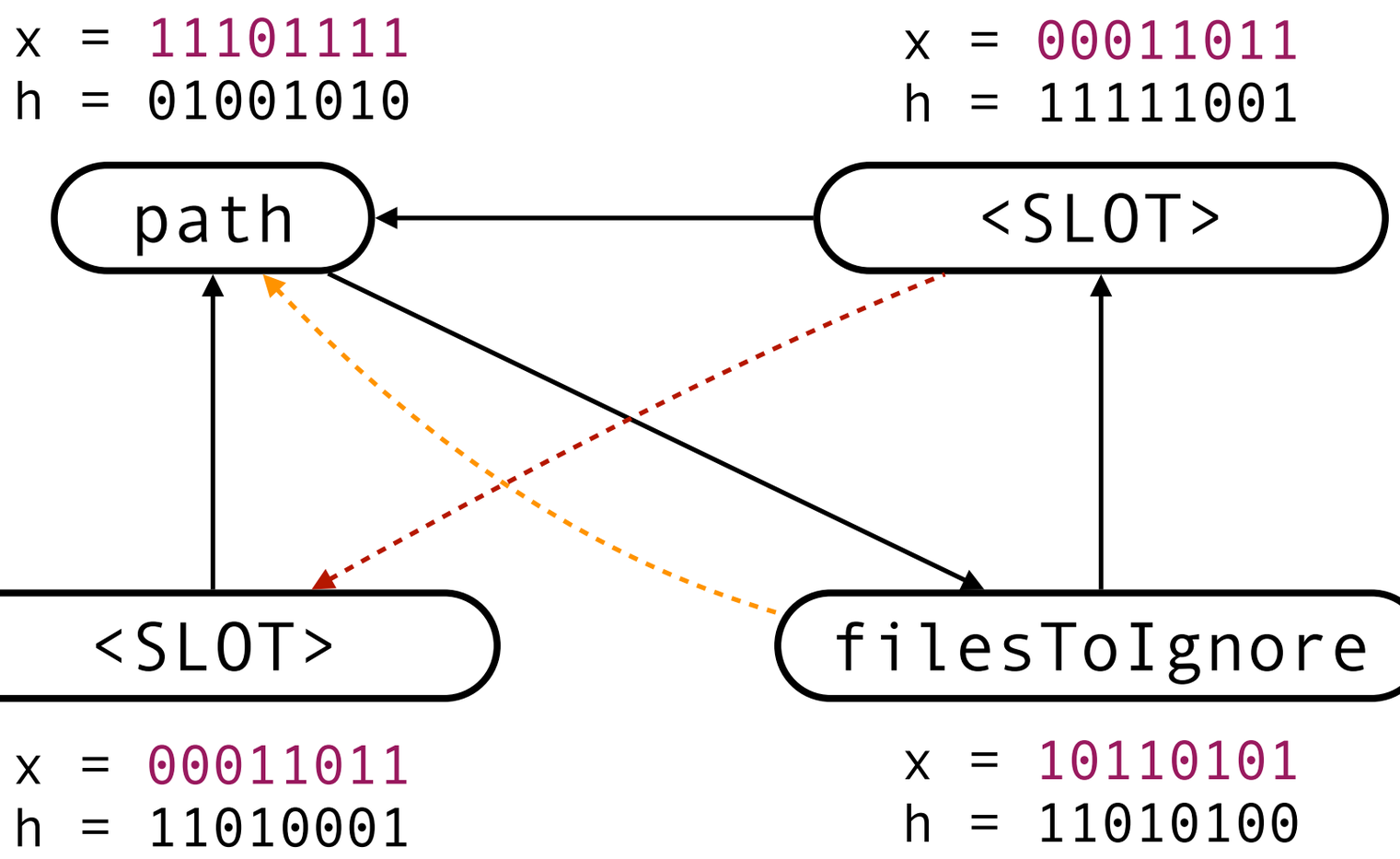

x = 11101111
h = 10101011
x = 11011101
h = 00010101
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11010101
x = 10110101
h = 00100100

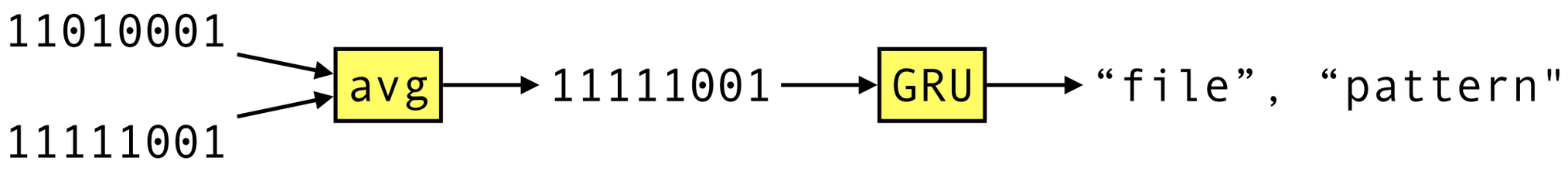

x = 11101111
h = 01001010
x = 11011101
h = 11111001
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11010001
x = 10110101
h = 11010100

VarNaming

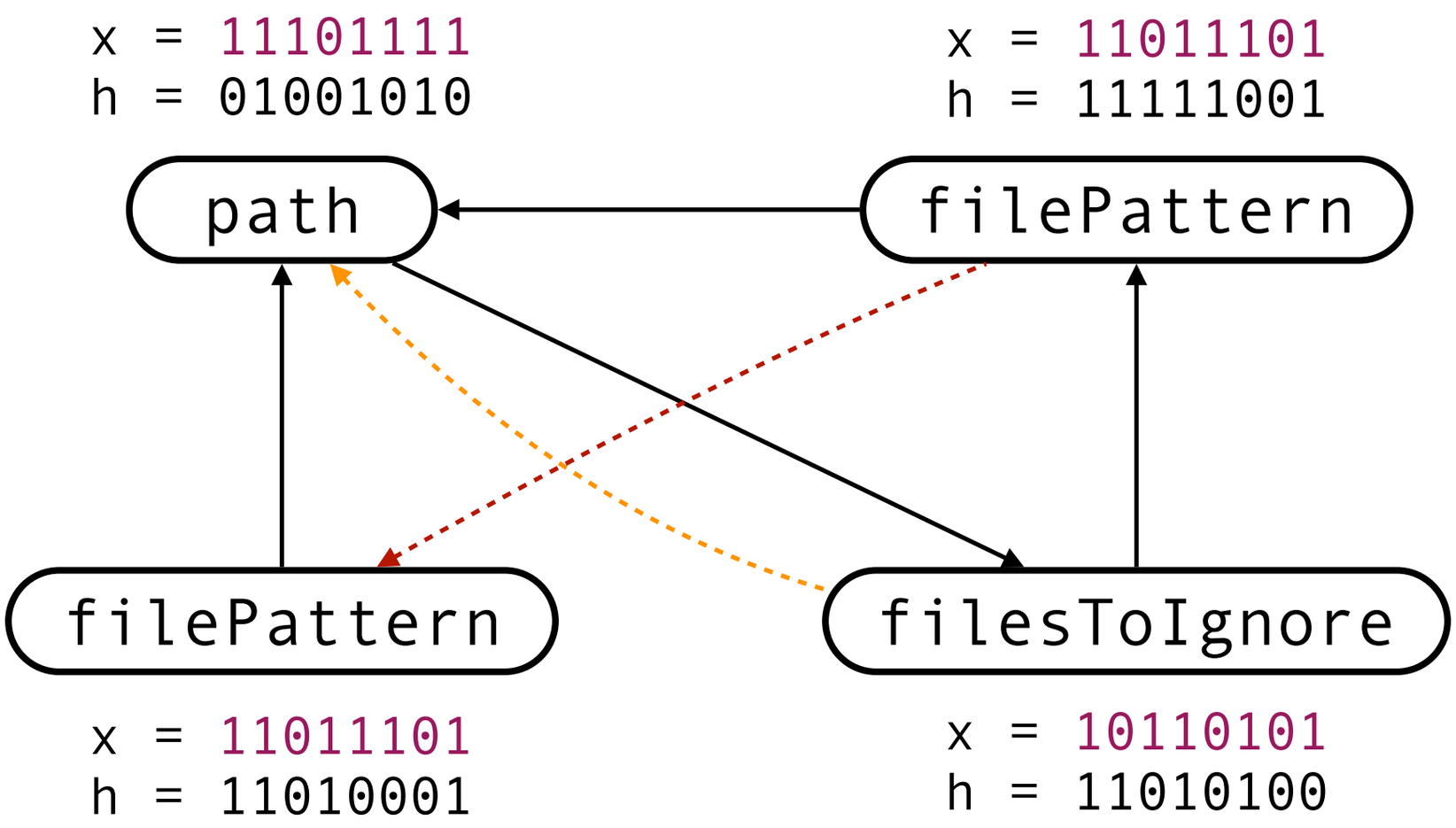

x = 11101111
h = 01001010
x = 00011011
h = 11111001
path
<SLOT>
<SLOT>
filesToIgnore
x = 00011011
h = 11010001
x = 10110101
h = 11010100

VarNaming

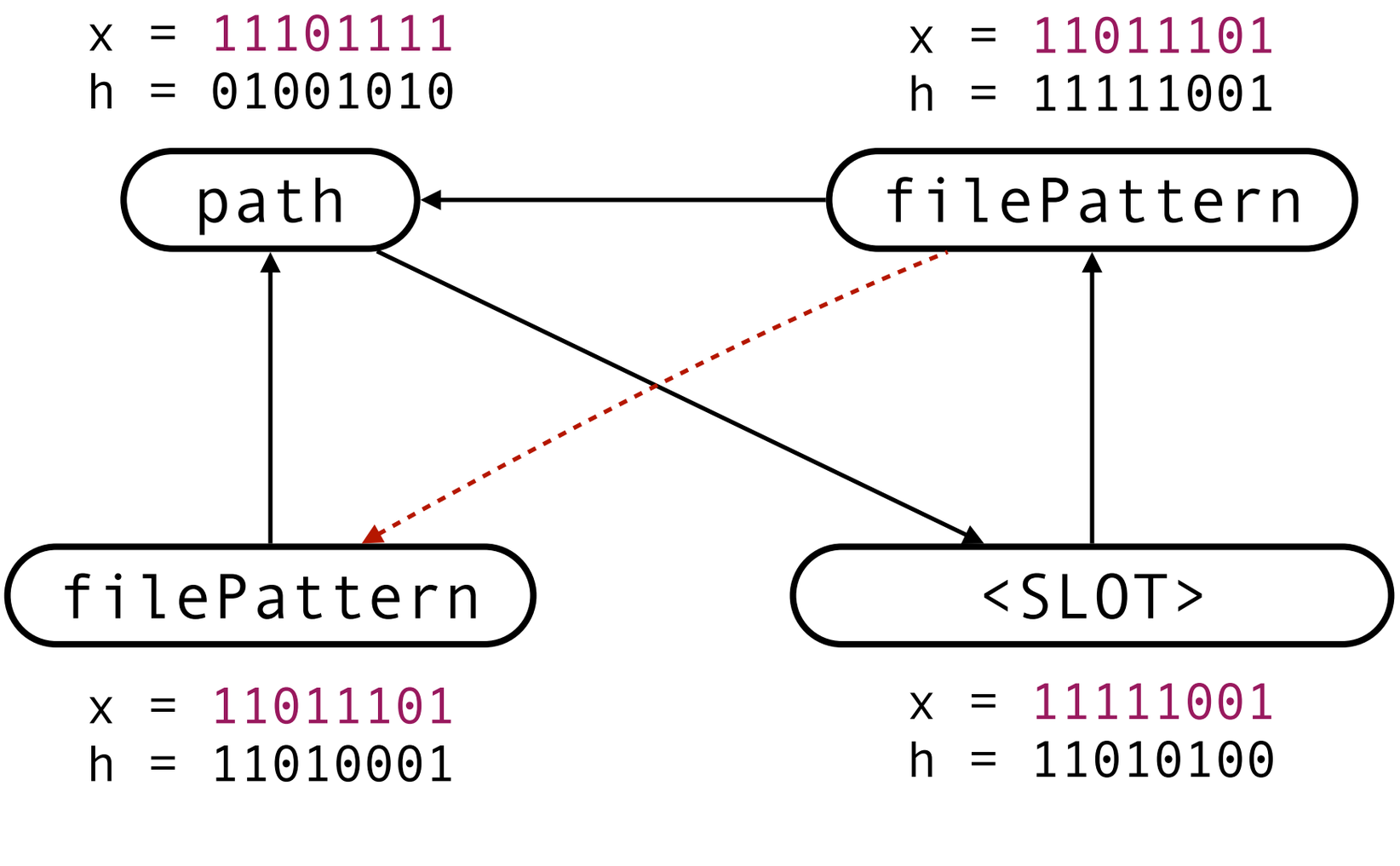

x = 11101111
h = 01001010
x = 00011011
h = 11111001
path
<SLOT>
<SLOT>
filesToIgnore
x = 00011011
h = 11010001
x = 10110101
h = 11010100


11010001
11111001
avg
11111001
GRU
“file”, “pattern"

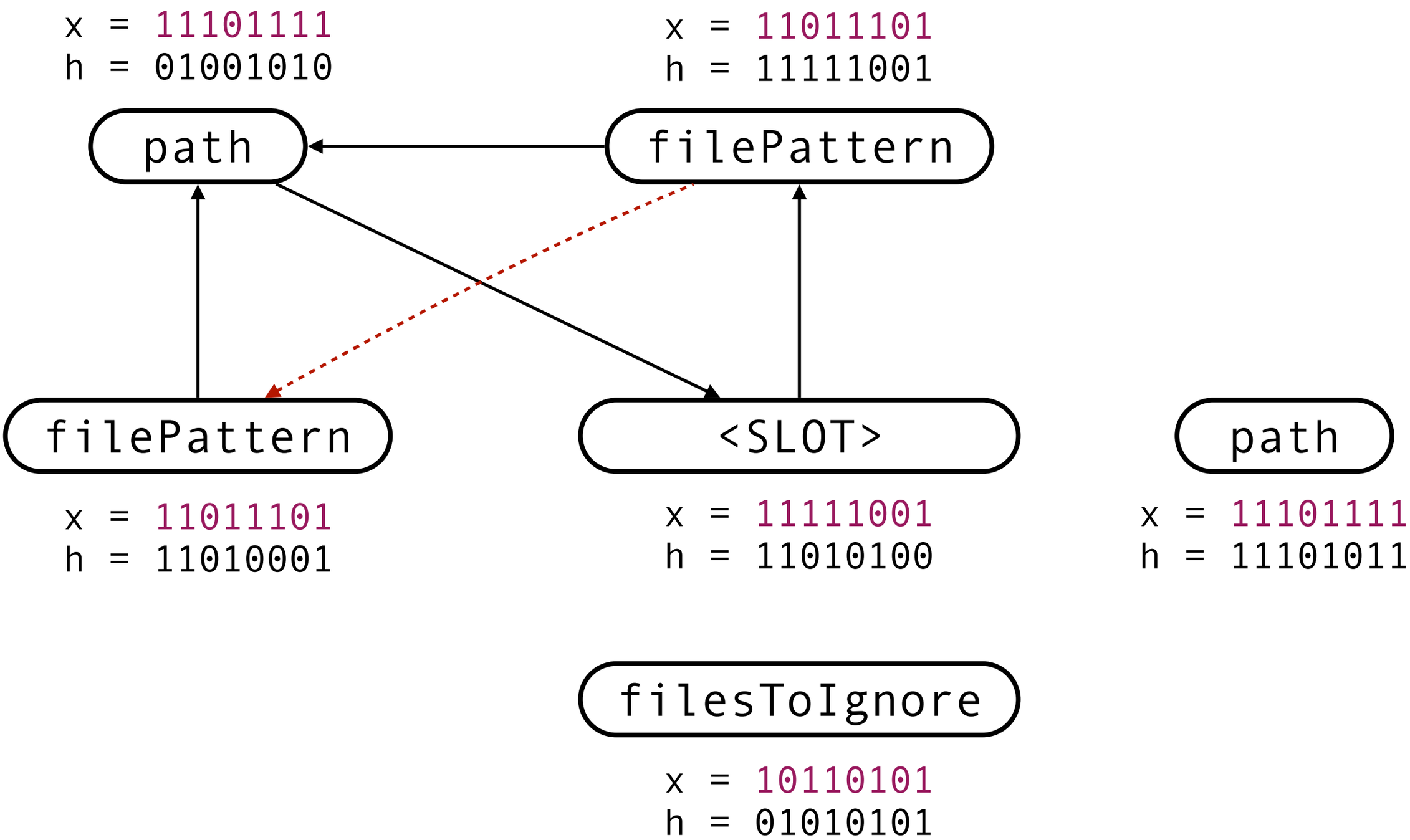
VarMisuse


x = 11101111
h = 01001010
x = 11011101
h = 11111001
path
filePattern
filePattern
filesToIgnore
x = 11011101
h = 11010001
x = 10110101
h = 11010100

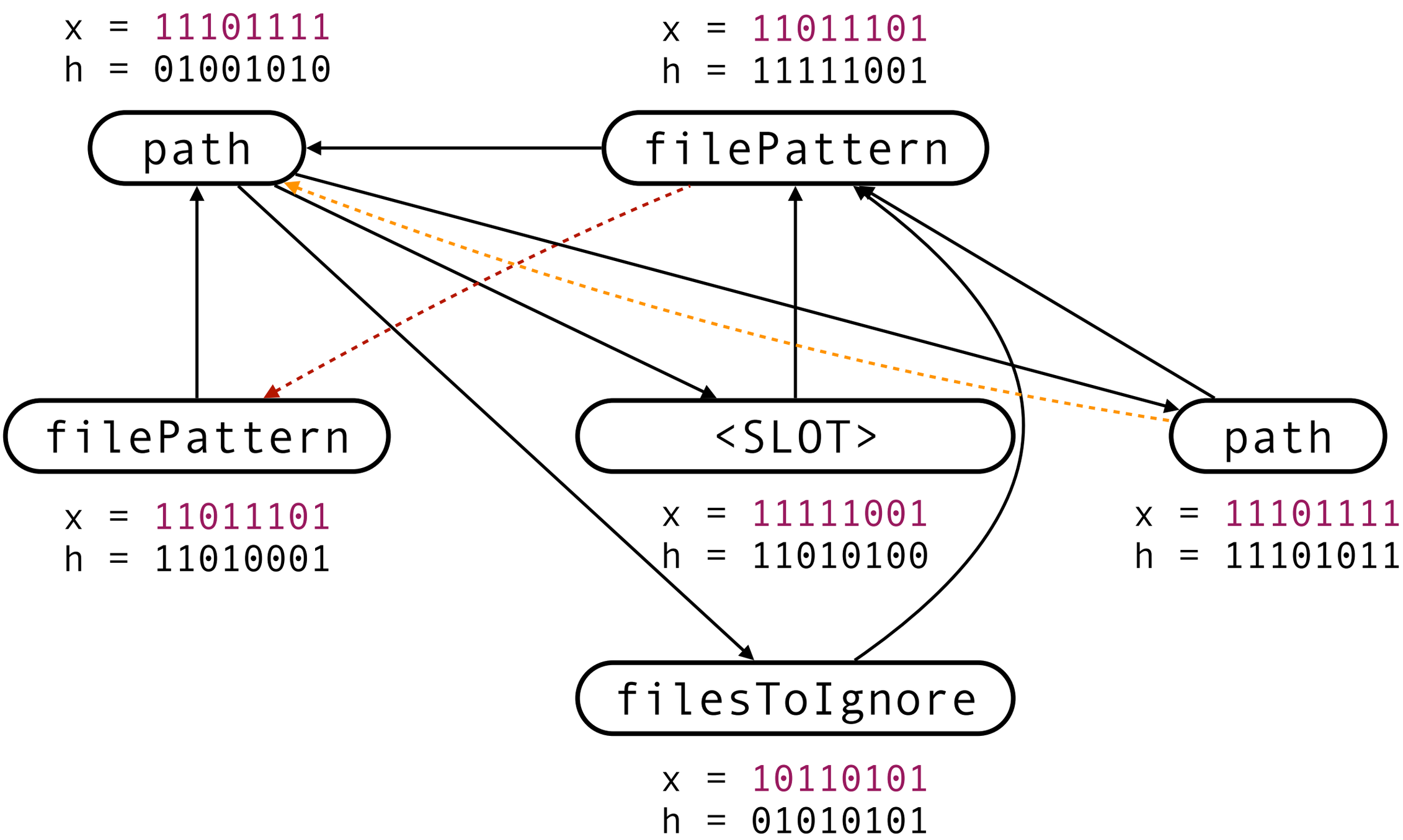
VarMisuse


x = 11101111
h = 01001010
x = 11011101
h = 11111001
path
filePattern
filePattern
<SLOT>
x = 11011101
h = 11010001
x = 11111001
h = 11010100

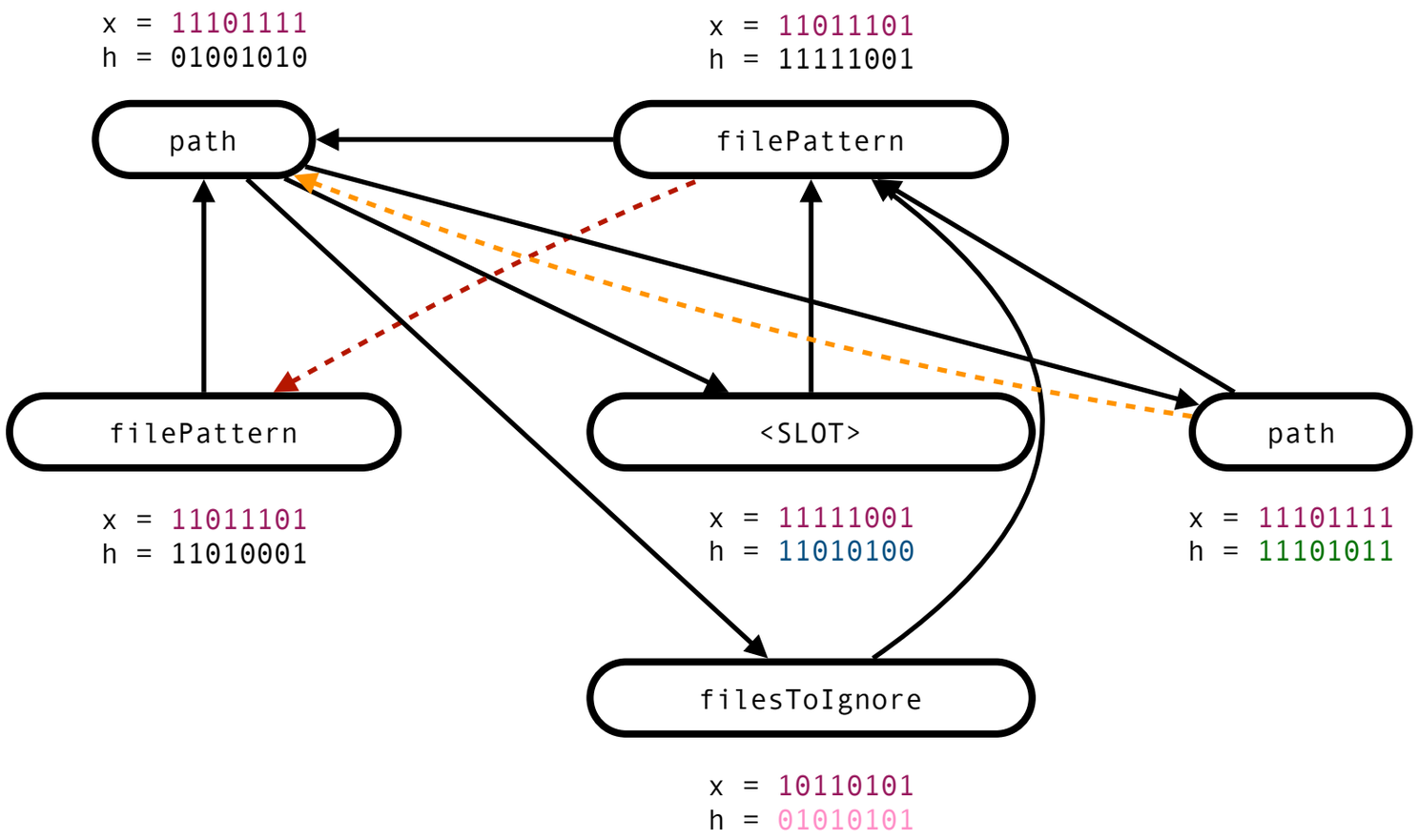
VarMisuse


x = 11101111
h = 01001010
x = 11011101
h = 11111001
path
filePattern
filePattern
x = 11011101
h = 11010001
<SLOT>
x = 11111001
h = 11010100
path
x = 11101111
h = 11101011
filesToIgnore
x = 10110101
h = 01010101

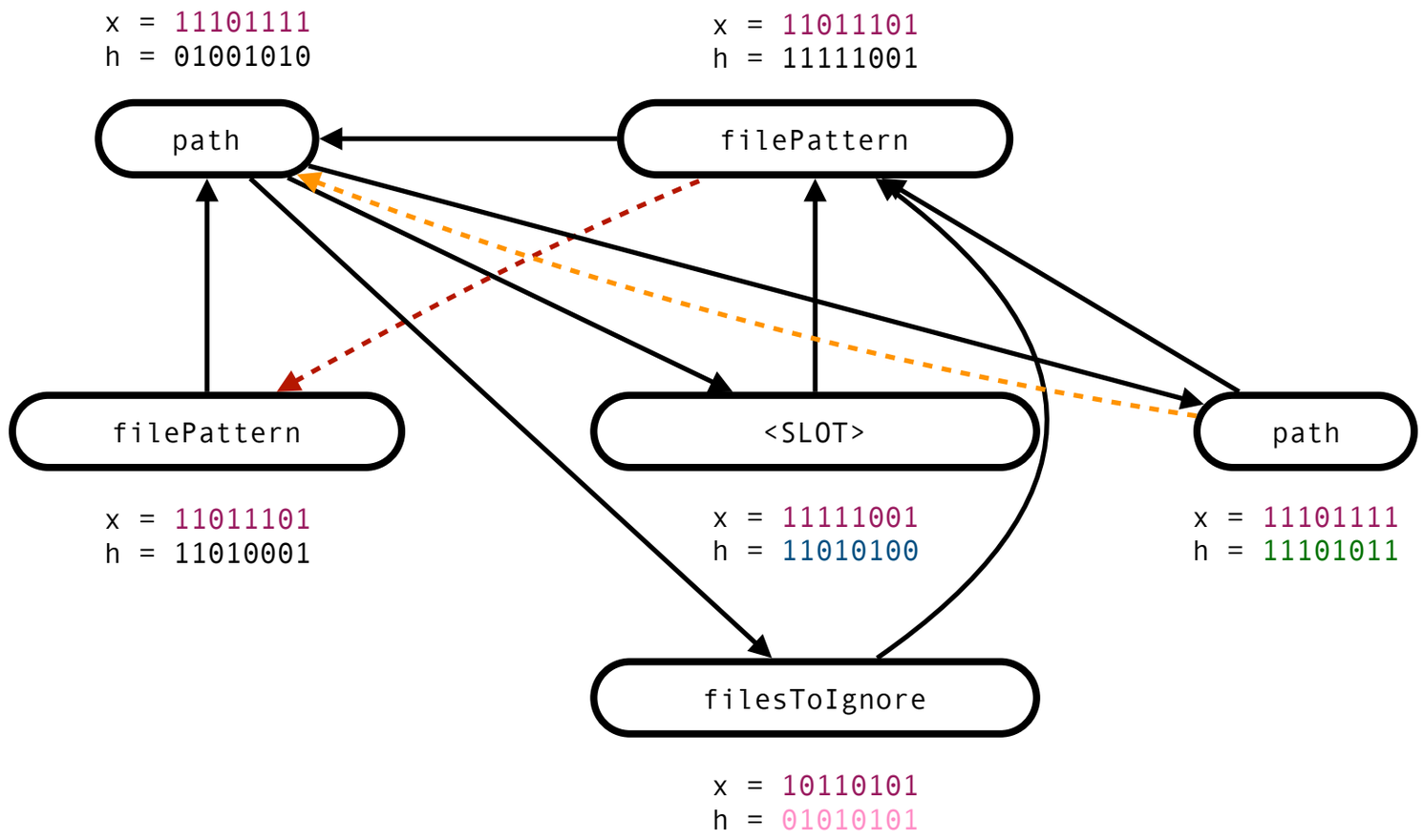
VarMisuse

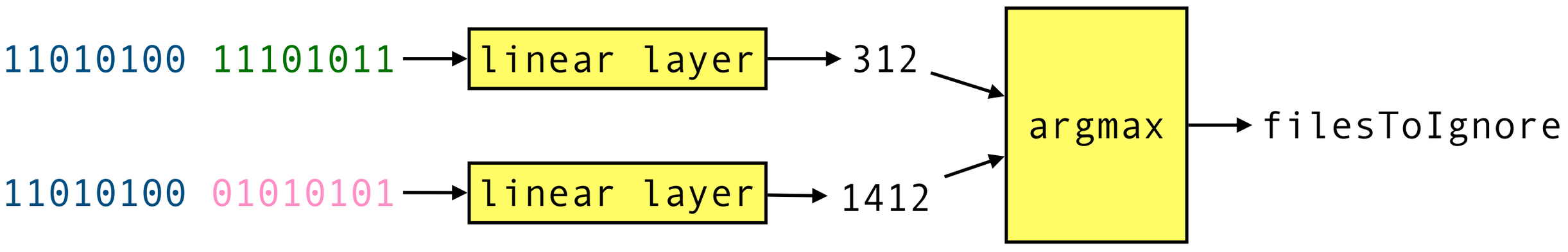

x = 11101111
h = 01001010
x = 11011101
h = 11111001
path
filePattern
filePattern
x = 11011101
h = 11010001
<SLOT>
x = 11111001
h = 11010100
path
x = 11101111
h = 11101011
filesToIgnore
x = 10110101
h = 01010101

VarMisuse


x = 11101111
h = 01001010
x = 11011101
h = 11111001
path
filePattern
filePattern
<SLOT>
path
x = 11011101
h = 11010001
x = 11111001
h = 11010100
x = 11101111
h = 11101011
filesToIgnore
x = 10110101
h = 01010101

VarMisuse


x = 11101111
h = 01001010
x = 11011101
h = 11111001
path
filePattern
filePattern
<SLOT>
path
x = 11011101
h = 11010001
x = 11111001
h = 11010100
x = 11101111
h = 11101011
filesToIgnore
x = 10110101
h = 01010101


11010100 11101011
linear layer
312
11010100 01010101
linear layer
1412
argmax
filesToIgnore

Previous Work

- Treat programs as sequences of tokens
- Treat programs as ASTs without fancy edges

Previous Work

This is not true. The whole point of many recent works is exactly to not learn over shallow syntactic representations but to leverage semantic information. For example, [1][2] introduce semantic relations between program elements (including data-flow, e.g., initialized-by, read-before, wrote-before, and others), [3,8] use semantic analysis to extract sequences of method calls on a given object, [4,5] use both structural dependencies extracted from AST and data dependencies computed via semantic analysis, graph based approaches such as [7], etc. [6] even tries to learn such semantic dependencies automatically instead of providing it by hand as part of the model.

Previous Work

Thank you for reading our work and for your comments. First, let us point out that some of the papers you note as missing are discussed in our submission (namely, [2,3,6] in your notation, which we felt to be the most influential contributions in the field). Due to the page size limit, we had to make hard decisions which related work to highlight. We understand that your opinion here differs, and we will try to take it into account when preparing future versions. We refer readers interested in the overall field to the https://ml4code.github.io effort, whose focus is a literature review.

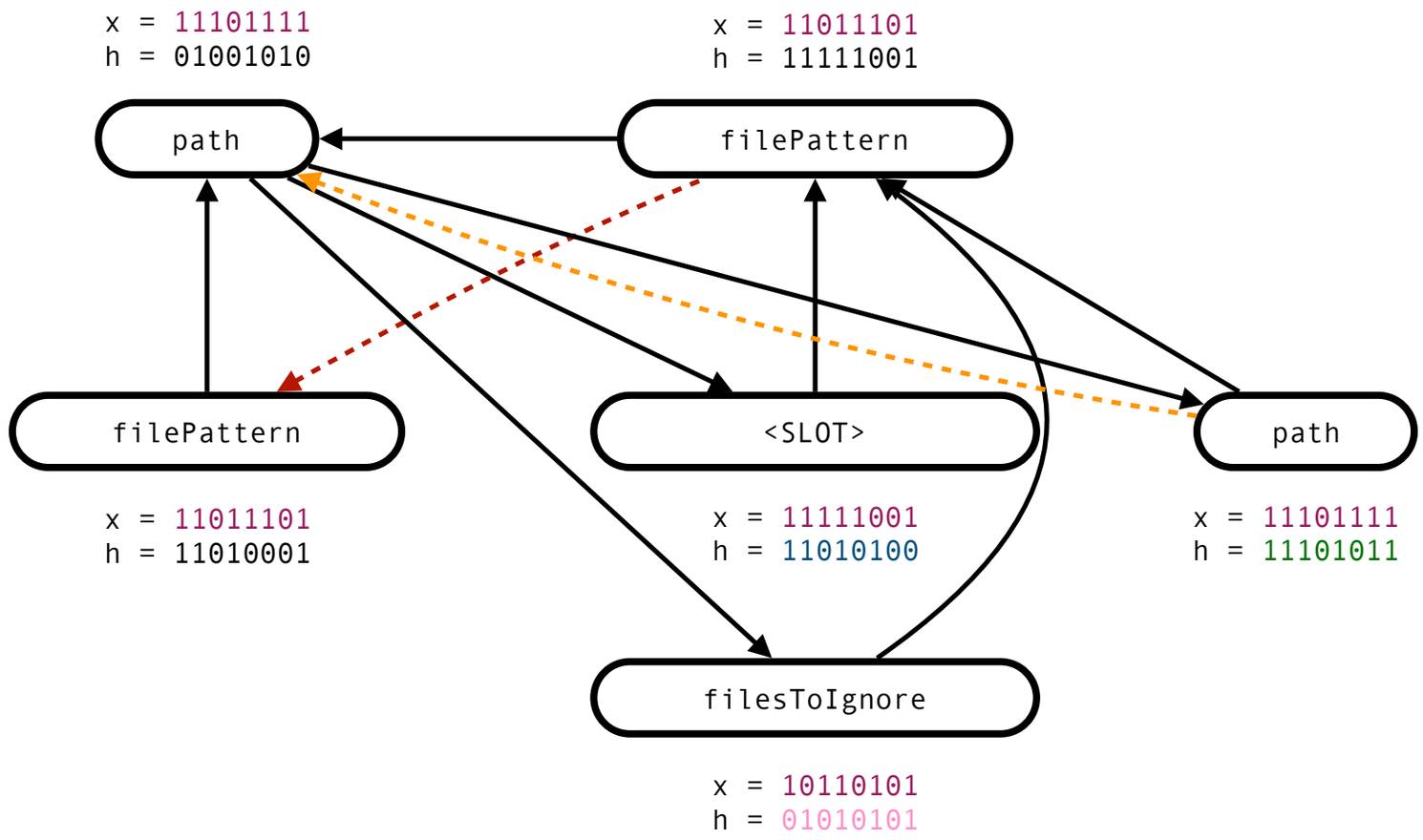
Name	Git SHA	kLOCs	Slots	Vars	Description
Akka.NET	719335a1	240	51.3k	51.2k	Actor-based Concurrent & Distributed Framework
AutoMapper	2ca7c2b5	46	3.7k	10.7k	Object-to-Object Mapping Library
BenchmarkDotNet	1670ca34	28	5.1k	6.1k	Benchmarking Library
BotBuilder	190117c3	44	6.4k	8.7k	SDK for Building Bots
choco	93985688	36	3.8k	5.2k	Windows Package Manager
commandline†	09677b16	11	1.1k	2.3k	Command Line Parser
CommonMark.NET^{Dev}	f3d54530	14	2.6k	1.4k	Markdown Parser
Dapper	931c700d	18	3.3k	4.7k	Object Mapper Library
EntityFramework	fa0b7ec8	263	33.4k	39.3k	Object-Relational Mapper
Hangfire	ffc4912f	33	3.6k	6.1k	Background Job Processing Library
Humanizer†	cc11a77e	27	2.4k	4.4k	String Manipulation and Formatting
Lean†	f574bfd7	190	26.4k	28.3k	Algorithmic Trading Engine
Nancy	72e1f614	70	7.5k	15.7	HTTP Service Framework
Newtonsoft.Json	6057d9b8	123	14.9k	16.1k	JSON Library
Ninject	7006297f	13	0.7k	2.1k	Code Injection Library
NLog	643e326a	75	8.3k	11.0k	Logging Library
Opserver	51b032e7	24	3.7k	4.5k	Monitoring System
OptiKey	7d35c718	34	6.1k	3.9k	Assistive On-Screen Keyboard
orleans	e0d6a150	300	30.7k	35.6k	Distributed Virtual Actor Model
Polly	0afdbc32	32	3.8k	9.1k	Resilience & Transient Fault Handling Library
quartznet	b33e6f86	49	9.6k	9.8k	Scheduler
ravendb^{Dev}	55230922	647	78.0k	82.7k	Document Database
RestSharp	70de357b	20	4.0k	4.5k	REST and HTTP API Client Library
Rx.NET	2d146fe5	180	14.0k	21.9k	Reactive Language Extensions
scriptcs	f3cc8bcb	18	2.7k	4.3k	C# Text Editor
ServiceStack	6d59da75	231	38.0k	46.2k	Web Framework
ShareX	718dd711	125	22.3k	18.1k	Sharing Application
SignalR	fa88089e	53	6.5k	10.5k	Push Notification Framework
Wox	cdaf6272	13	2.0k	2.1k	Application Launcher

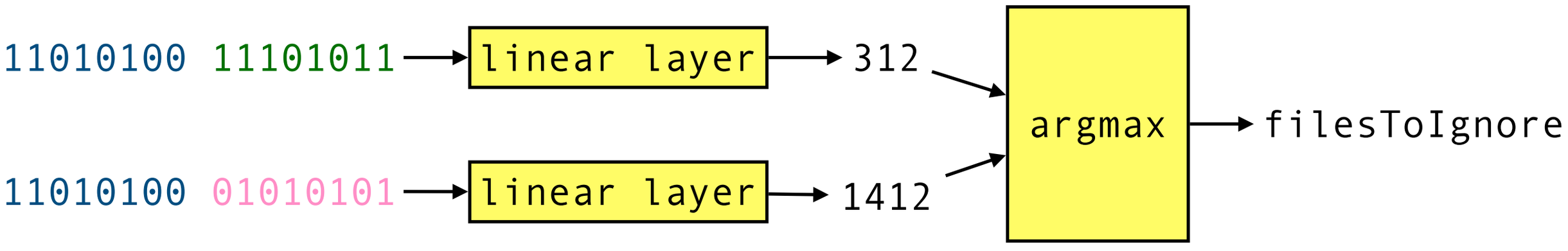
Name	Git SHA	kLOCs	Slots	Vars	Description
Akka.NET	719335a1	240	51.3k	51.2k	Actor-based Concurrent & Distributed Framework
AutoMapper	2ca7c2b5	46	3.7k	10.7k	Object-to-Object Mapping Library
BenchmarkDotNet	1670ca34	28	5.1k	6.1k	Benchmarking Library
BotBuilder	190117c3	44	6.4k	8.7k	SDK for Building Bots
choco	93985688	36	3.8k	5.2k	Windows Package Manager
commandline†	09677b16	11	1.1k	2.3k	Command Line Parser
CommonMark.NET^{Dev}	f3d54530	14	2.6k	1.4k	Markdown Parser
Dapper	931c700d	18	3.3k	4.7k	Object Mapper Library
EntityFramework	fa0b7ec8	263	33.4k	39.3k	Object-Relational Mapper
Hangfire	ffc4912f	33	3.6k	6.1k	Background Job Processing Library
Humanizer†	cc11a77e	27	2.4k	4.4k	String Manipulation and Formatting
Lean†	f574bfd7	190	26.4k	28.3k	Algorithmic Trading Engine
Nancy	72e1f614	70	7.5k	15.7	HTTP Service Framework
Newtonsoft.Json	6057d9b8	123	14.9k	16.1k	JSON Library
Ninject	7006297f	13	0.7k	2.1k	Code Injection Library
NLog	643e326a	75	8.3k	11.0k	Logging Library
Opserver	51b032e7	24	3.7k	4.5k	Monitoring System
OptiKey	7d35c718	34	6.1k	3.9k	Assistive On-Screen Keyboard
orleans	e0d6a150	300	30.7k	35.6k	Distributed Virtual Actor Model
Polly	0afdbc32	32	3.8k	9.1k	Resilience & Transient Fault Handling Library
quartznet	b33e6f86	49	9.6k	9.8k	Scheduler
ravendb^{Dev}	55230922	647	78.0k	82.7k	Document Database
RestSharp	70de357b	20	4.0k	4.5k	REST and HTTP API Client Library
Rx.NET	2d146fe5	180	14.0k	21.9k	Reactive Language Extensions
scriptcs	f3cc8bcb	18	2.7k	4.3k	C# Text Editor
ServiceStack	6d59da75	231	38.0k	46.2k	Web Framework
ShareX	718dd711	125	22.3k	18.1k	Sharing Application
SignalR	fa88089e	53	6.5k	10.5k	Push Notification Framework
Wox	cdaf6272	13	2.0k	2.1k	Application Launcher

Name	Git SHA	kLOCs	Slots	Vars	Description
Akka.NET	719335a1	240	51.3k	51.2k	Actor-based Concurrent & Distributed Framework
AutoMapper	2ca7c2b5	46	3.7k	10.7k	Object-to-Object Mapping Library
BenchmarkDotNet	1670ca34	28	5.1k	6.1k	Benchmarking Library
BotBuilder	190117c3	44	6.4k	8.7k	SDK for Building Bots
choco	93985688	36	3.8k	5.2k	Windows Package Manager
commandline†	09677b16	11	1.1k	2.3k	Command Line Parser
CommonMark.NET^{Dev}	f3d54530	14	2.6k	1.4k	Markdown Parser
Dapper	931c700d	18	3.3k	4.7k	Object Mapper Library
EntityFramework	fa0b7ec8	263	33.4k	39.3k	Object-Relational Mapper
Hangfire	ffc4912f	33	3.6k	6.1k	Background Job Processing Library
Humanizer†	cc11a77e	27	2.4k	4.4k	String Manipulation and Formatting
Lean†	f574bfd7	190	26.4k	28.3k	Algorithmic Trading Engine
Nancy	72e1f614	70	7.5k	15.7	HTTP Service Framework
Newtonsoft.Json	6057d9b8	123	14.9k	16.1k	JSON Library
Ninject	7006297f	13	0.7k	2.1k	Code Injection Library
NLog	643e326a	75	8.3k	11.0k	Logging Library
Opserver	51b032e7	24	3.7k	4.5k	Monitoring System
OptiKey	7d35c718	34	6.1k	3.9k	Assistive On-Screen Keyboard
orleans	e0d6a150	300	30.7k	35.6k	Distributed Virtual Actor Model
Polly	0afdbc32	32	3.8k	9.1k	Resilience & Transient Fault Handling Library
quartznet	b33e6f86	49	9.6k	9.8k	Scheduler
ravendb^{Dev}	55230922	647	78.0k	82.7k	Document Database
RestSharp	70de357b	20	4.0k	4.5k	REST and HTTP API Client Library
Rx.NET	2d146fe5	180	14.0k	21.9k	Reactive Language Extensions
scriptcs	f3cc8bcb	18	2.7k	4.3k	C# Text Editor
ServiceStack	6d59da75	231	38.0k	46.2k	Web Framework
ShareX	718dd711	125	22.3k	18.1k	Sharing Application
SignalR	fa88089e	53	6.5k	10.5k	Push Notification Framework
Wox	cdaf6272	13	2.0k	2.1k	Application Launcher

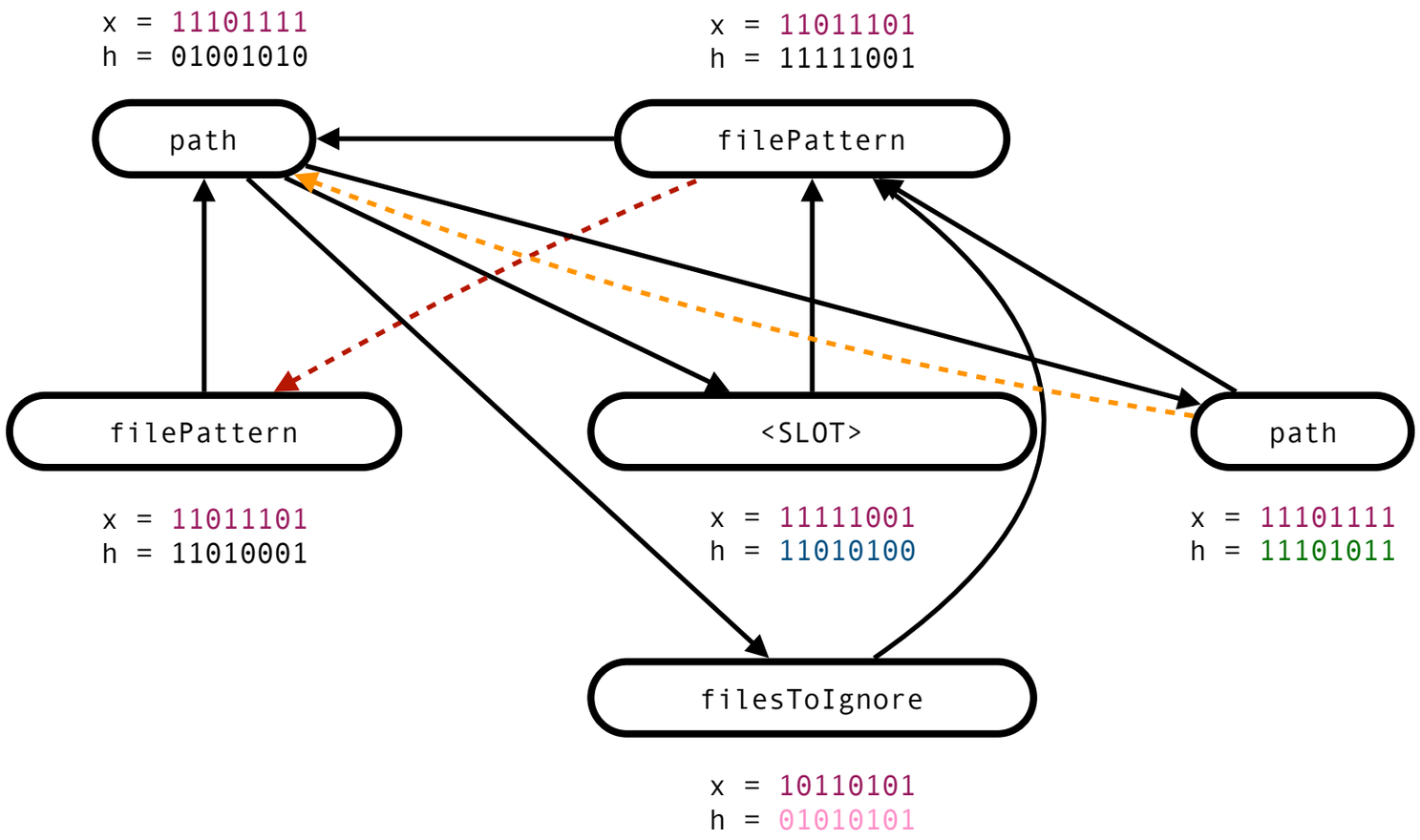
Name	Git SHA	kLOCs	Slots	Vars	Description
Akka.NET	719335a1	240	51.3k	51.2k	Actor-based Concurrent & Distributed Framework
AutoMapper	2ca7c2b5	46	3.7k	10.7k	Object-to-Object Mapping Library
BenchmarkDotNet	1670ca34	28	5.1k	6.1k	Benchmarking Library
BotBuilder	190117c3	44	6.4k	8.7k	SDK for Building Bots
choco	93985688	36	3.8k	5.2k	Windows Package Manager
commandline†	09677b16	11	1.1k	2.3k	Command Line Parser
CommonMark.NET^{Dev}	f3d54530	14	2.6k	1.4k	Markdown Parser
Dapper	931c700d	18	3.3k	4.7k	Object Mapper Library
EntityFramework	fa0b7ec8	263	33.4k	39.3k	Object-Relational Mapper
Hangfire	ffc4912f	33	3.6k	6.1k	Background Job Processing Library
Humanizer†	cc11a77e	27	2.4k	4.4k	String Manipulation and Formatting
Lean†	f574bfd7	190	26.4k	28.3k	Algorithmic Trading Engine
Nancy	72e1f614	70	7.5k	15.7	HTTP Service Framework
Newtonsoft.Json	6057d9b8	123	14.9k	16.1k	JSON Library
Ninject	7006297f	13	0.7k	2.1k	Code Injection Library
NLog	643e326a	75	8.3k	11.0k	Logging Library
Opserver	51b032e7	24	3.7k	4.5k	Monitoring System
OptiKey	7d35c718	34	6.1k	3.9k	Assistive On-Screen Keyboard
orleans	e0d6a150	300	30.7k	35.6k	Distributed Virtual Actor Model
Polly	0afdbc32	32	3.8k	9.1k	Resilience & Transient Fault Handling Library
quartznet	b33e6f86	49	9.6k	9.8k	Scheduler
ravendb^{Dev}	55230922	647	78.0k	82.7k	Document Database
RestSharp	70de357b	20	4.0k	4.5k	REST and HTTP API Client Library
Rx.NET	2d146fe5	180	14.0k	21.9k	Reactive Language Extensions
scriptcs	f3cc8bcb	18	2.7k	4.3k	C# Text Editor
ServiceStack	6d59da75	231	38.0k	46.2k	Web Framework
ShareX	718dd711	125	22.3k	18.1k	Sharing Application
SignalR	fa88089e	53	6.5k	10.5k	Push Notification Framework
Wox	cdaf6272	13	2.0k	2.1k	Application Launcher

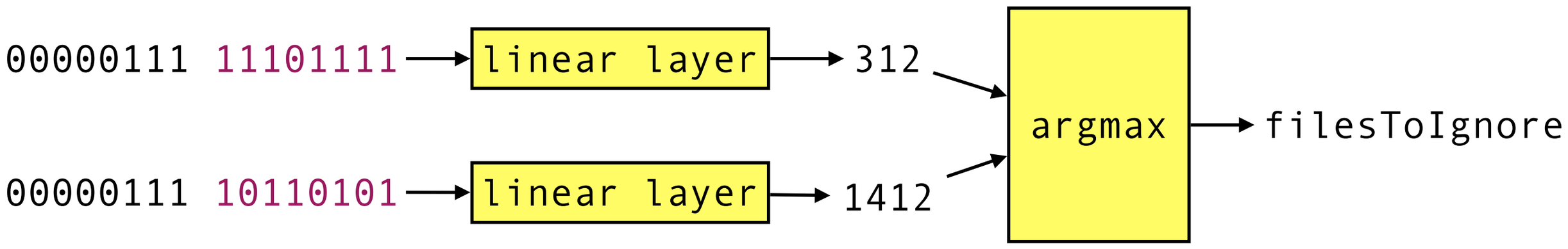
VarMisuse: Loc





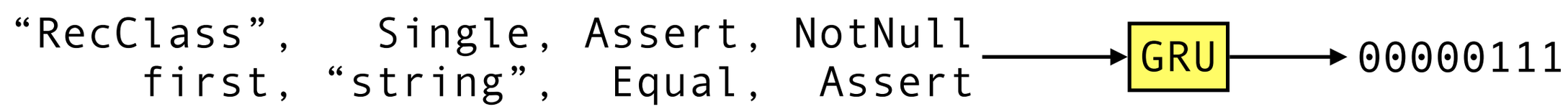
VarMisuse: Loc



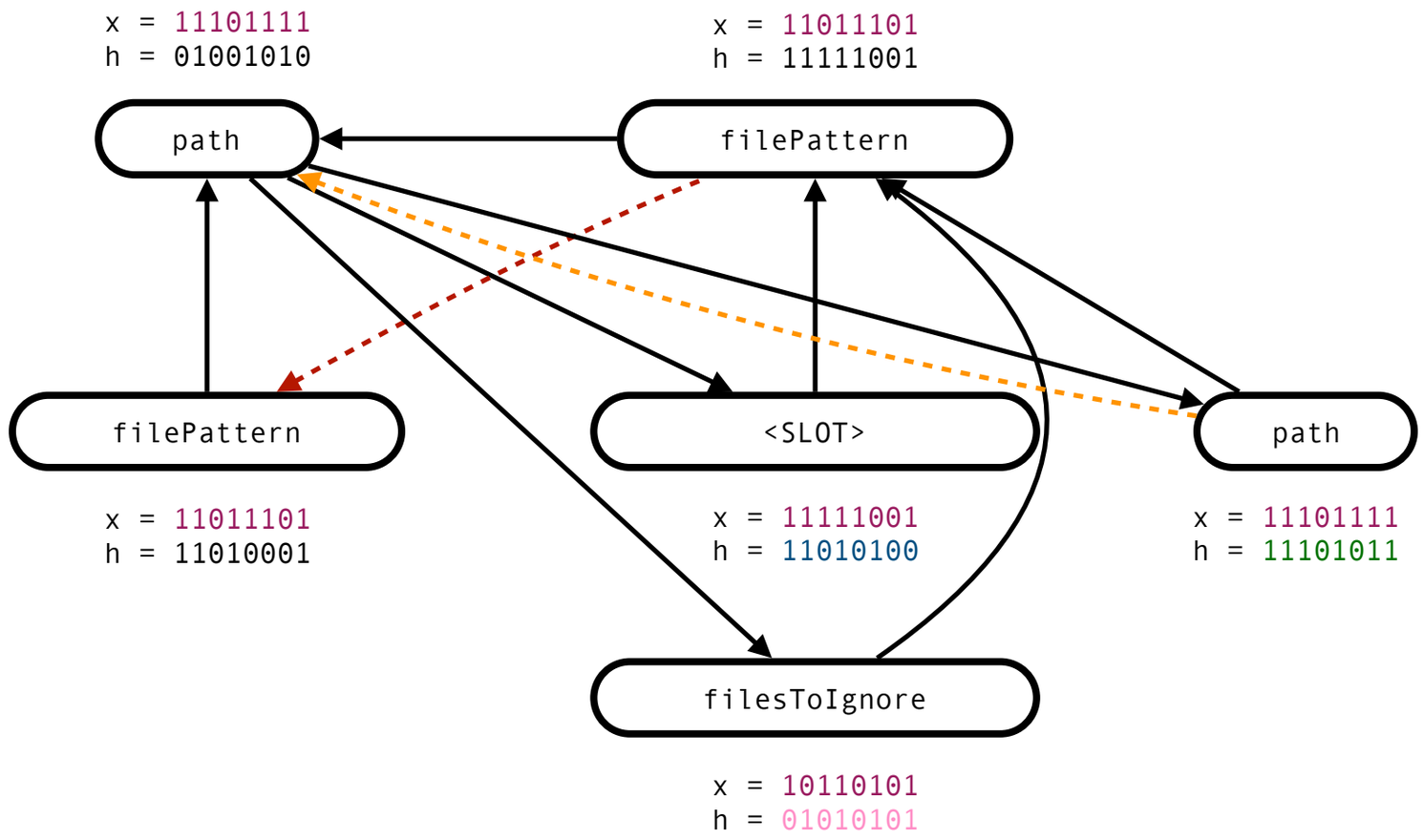


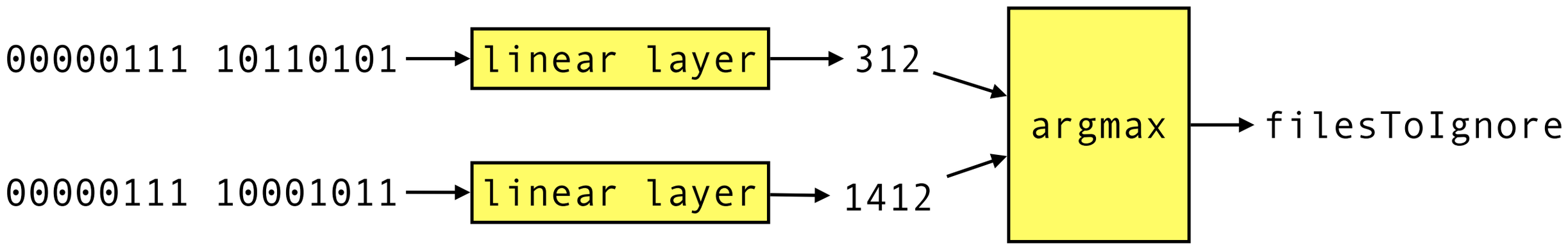
VarMisuse: Loc

```
var clazz = classTypes["Root"].Single();
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single();
Assert.NotNull(clazz);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```



VarMisuse: AvgBiRNN





VarMisuse: AvgBiRNN

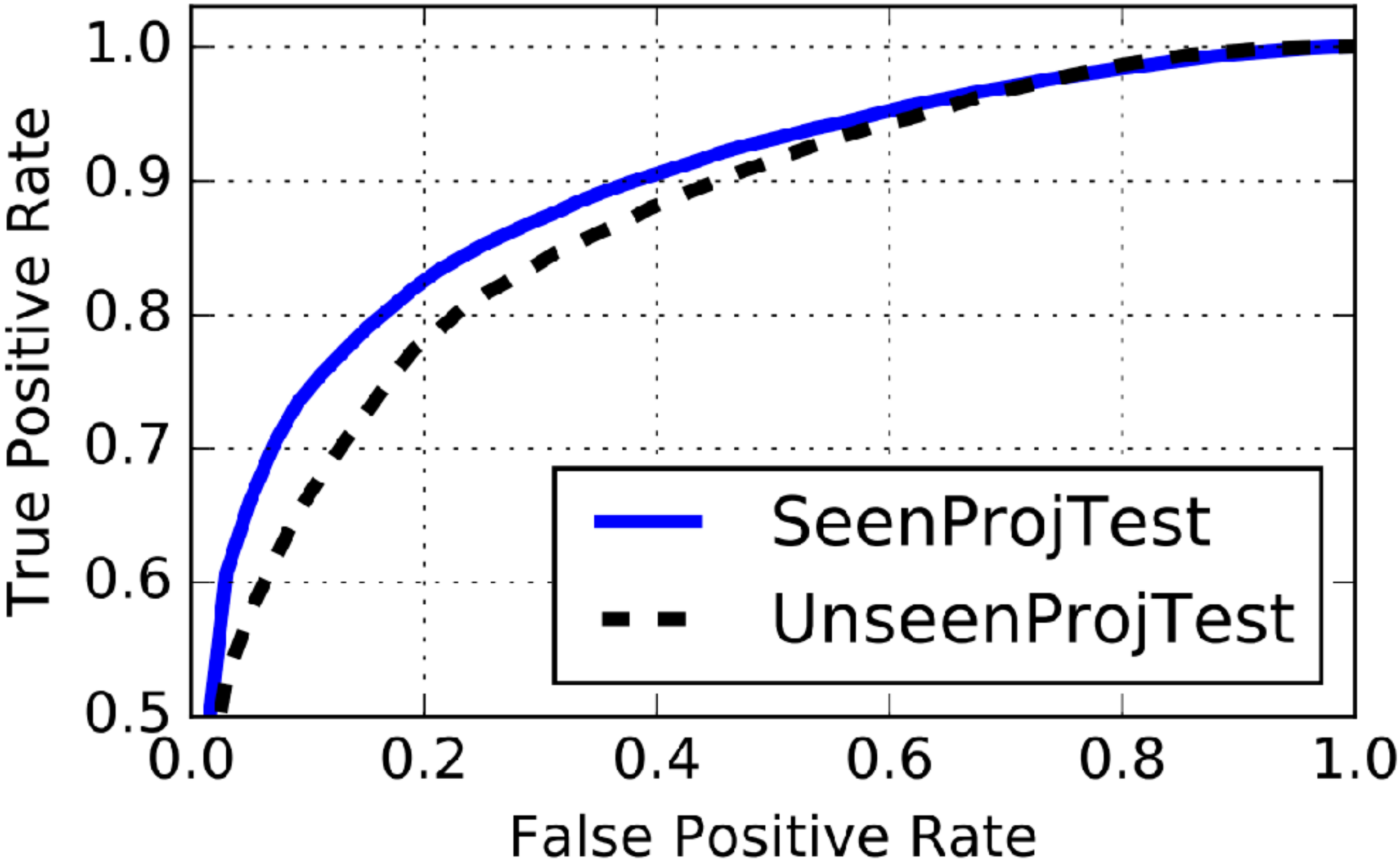
```
var clazz = classTypes["Root"].Single();
Assert.NotNull(clazz);
```

```
var first = classTypes["RecClass"].Single();
Assert.NotNull(clazz);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Mean: **3.8** variables
Median: **3** variables
Stddev: **2.6** variables

Table 1: Evaluation of models. SEENPROJTEST refers to the test set containing projects that have files in the training set, UNSEENPROJTEST refers to projects that have no files in the training data. Results averaged over two runs.

	SEENPROJTEST				UNSEENPROJTEST			
	LOC	AVGLBL	AVGBIRNN	GGNN	LOC	AVGLBL	AVGBIRNN	GGNN
VARMISUSE								
Accuracy (%)	50.0	—	73.7	**85.5**	28.9	—	60.2	**78.2**
PR AUC	0.788	—	0.941	**0.980**	0.611	—	0.895	**0.958**



(a) Precision-Recall Curve

VarNaming: AvgLBL, AvgBiRNN

```
import os

for root, dirs, files in os.walk("/mydir"):
    for [redacted] in files:
        if [redacted].endswith(".txt"):
            print(os.path.join(root, [redacted]))
```

Table 1: Evaluation of models. SEENPROJTEST refers to the test set containing projects that have files in the training set, UNSEENPROJTEST refers to projects that have no files in the training data. Results averaged over two runs.

	SEENPROJTEST				UNSEENPROJTEST			
	LOC	AVGLBL	AVGBIRNN	GGNN	LOC	AVGLBL	AVGBIRNN	GGNN
VARMISUSE								
Accuracy (%)	50.0	—	73.7	**85.5**	28.9	—	60.2	**78.2**
PR AUC	0.788	—	0.941	**0.980**	0.611	—	0.895	**0.958**
VARNAMING								
Accuracy (%)	—	36.1	42.9	**53.6**	—	22.7	23.4	**44.0**
F1 (%)	—	44.0	50.1	**65.8**	—	30.6	32.0	**62.0**

Evaluation

For the variable naming task, the state-of-the-art approach achieves 79.1% for Obfuscated Android applications [1] (source code available online at http://nice2predict.org/). In comparison, this work achieves accuracy 19.3% which is 4x lower. These results are however not even mentioned in the paper. Worse, prior work considers a more difficult task in which all program identifiers are initially unknown. In contrast, the task considered here renames each variable separately, while knowing the correct names of all other variables.

Evaluation

We do not compare directly to [1] because our VarRename task focuses on names of local variables in general C# applications, and thus our toolchain is not able to infer names for classes and packages in Android applications at this time. However, let us note that even rough comparisons across different datasets for this task are practically impossible: In internal tests with the variable naming task, we found the accuracy of the same model to vary between ~15% and ~65% on datasets extracted from different projects. Finally, we consider the naming of local variables, whereas the 79.1% accuracy you refer to considers fields, methods, classes and packages (but no local variables). We have reason to believe that the task on the Android App dataset is on the “easier” end of the spectrum, as it comes for a single domain with highly specific APIs, idioms and domain-specific vocabulary, whereas our dataset comes from a highly diverse set of projects including everything from algorithmic trading code to code injection libraries.

Table 2: Ablation study for the GGNN model on SEENPROJTEST for the two tasks.

Ablation Description	Accuracy (%) VARMISUSE	VARNAMING
Standard Model (reported in Table 1)	85.5	53.6
Only NextToken, Child, LastUse, LastWrite edges	80.6	31.2
Only semantic edges (all but NextToken, Child)	78.4	52.9
Only syntax edges (NextToken, Child)	55.3	34.3
Node Labels: Tokens instead of subtokens	85.6	34.5
Node Labels: Disabled	84.3	31.8

Manually reviewed 500 locations in RavenDB and Roslyn. Found three “bugs” each.

Discussion

- Are these tools good enough yet? Will they ever be good enough?
- Are other tools simpler and better (e.g., linters)?
- Are these tools even targeting the right bugs? How many bugs are caused by using the wrong variable as opposed to race conditions, forgetting to update a variable, null variable access, broken invariants, etc?
- Are there opportunities to use AI not for automatic bug finding but for assisted bug finding? Smart fuzzing? Log anomaly detection? Trace anomaly detection?