Interactive and Continuous Program Reasoning

Kihong Heo
University of Pennsylvania

(join work with Sulekha Kulkarni, Mukund Raghothaman,
Xujie Si, Mayur Naik)

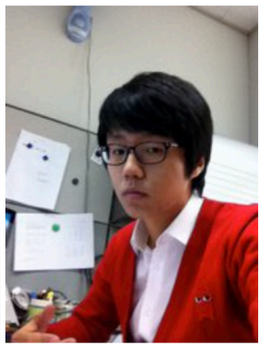
About Me

About Me

2018
(in the middle of postdoc)

2017
(at the end of Ph.D)

2015
(in the middle of Ph.D)



2011
(at the start of Ph.D)

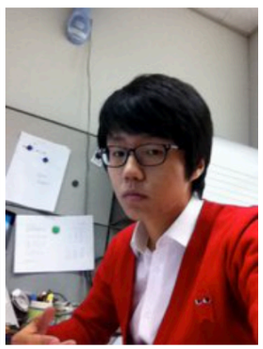
2009
(at the start of MS)

About Me

2018
(in the middle of postdoc)

2017
(at the end of Ph.D)

2015
(in the middle of Ph.D)



2011
(at the start of Ph.D)

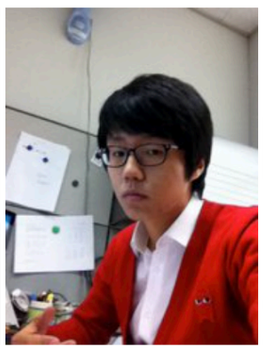
2009
(at the start of MS)

About Me

2018
(in the middle of postdoc)

2017
(at the end of Ph.D)

2015
(in the middle of Ph.D)



2011
(at the start of Ph.D)

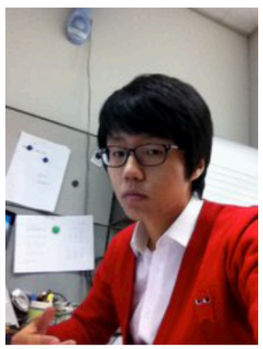
2009
(at the start of MS)

About Me

2018
(in the middle of postdoc)

2017
(at the end of Ph.D)

2015
(in the middle of Ph.D)



2011
(at the start of Ph.D)

2009
(at the start of MS)

About Me

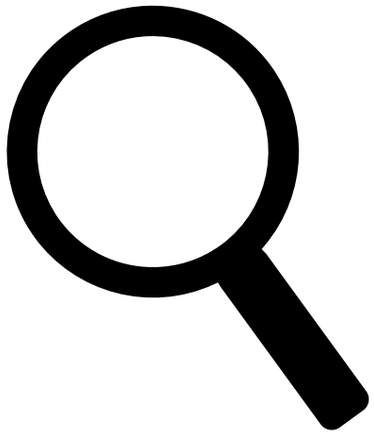
2018
(in the middle of postdoc)

2017
(at the end of Ph.D)

2015
(in the middle of Ph.D)

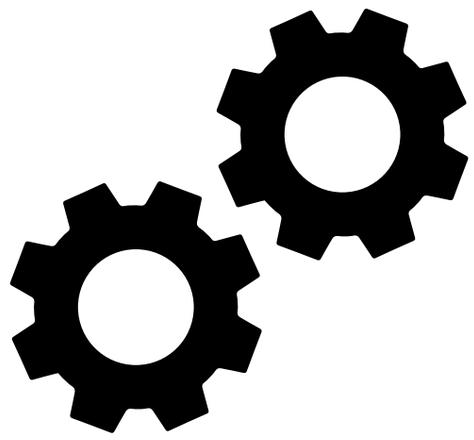
2011
(at the start of Ph.D)

2009
(at the start of MS)



Program Analysis [PLDI’18]

Program Debloating [CCS’18]



Program Synthesis [PLDI’18]

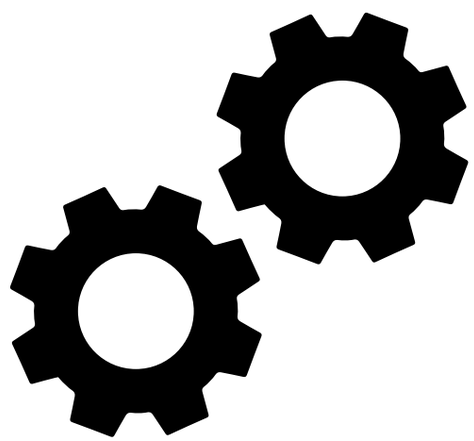
About Me

2018 (in the middle of postdoc) | 2017 (at the end of Ph.D) | 2015 (in the middle of Ph.D) | 2011 (at the start of Ph.D) | 2009 (at the start of MS)

Today's Talk

Program Analysis [PLDI'18]

Program Debloating [CCS'18]



Program Synthesis [PLDI'18]

Conventional Program Analysis

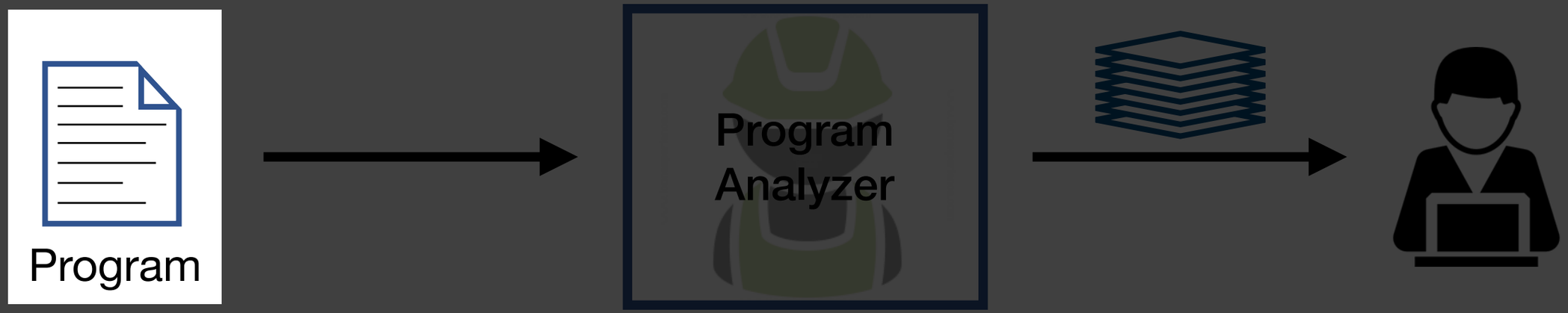
Conventional Program Analysis

1. Inflexible

Conventional Program Analysis

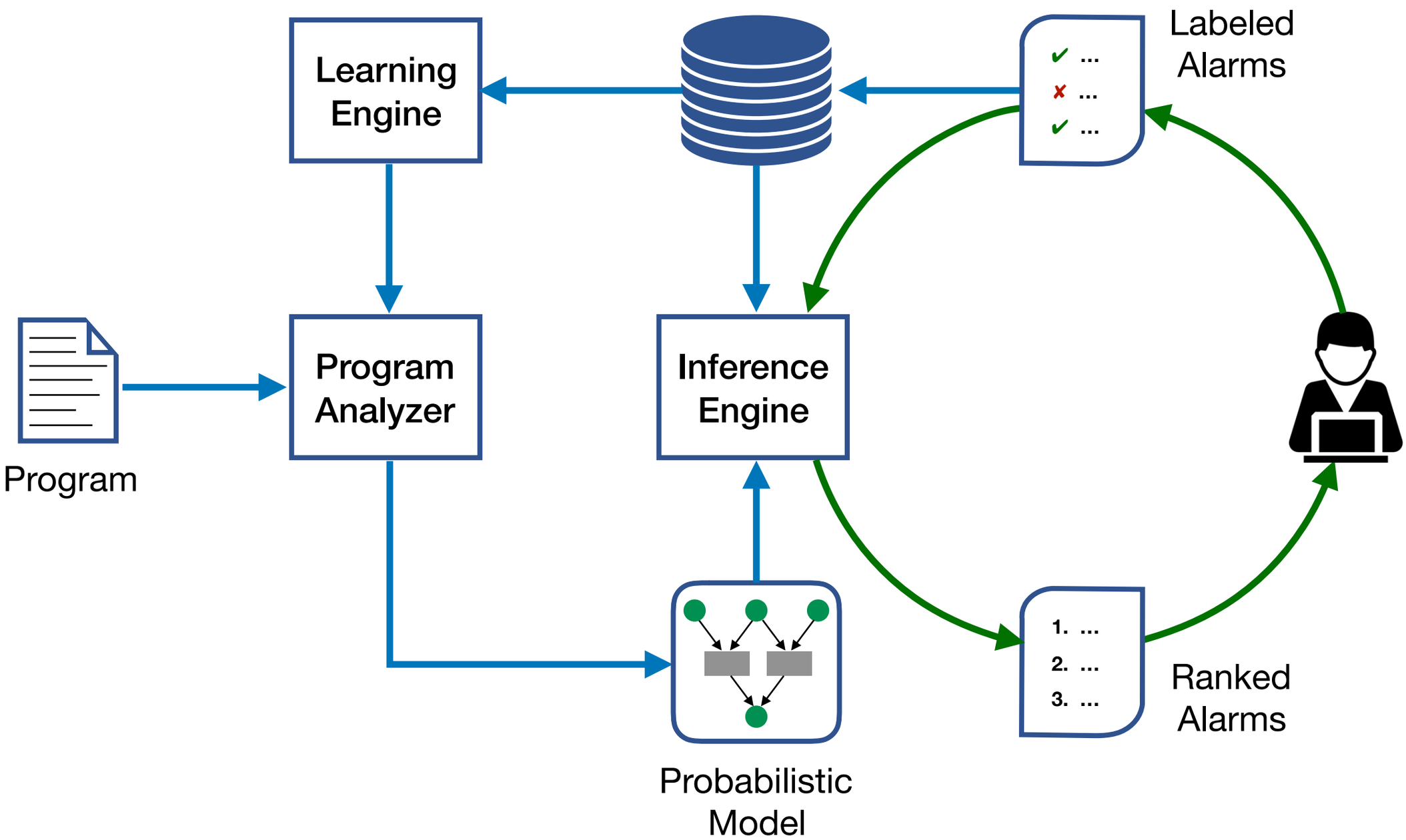
2. Unidirectional

Conventional Program Analysis

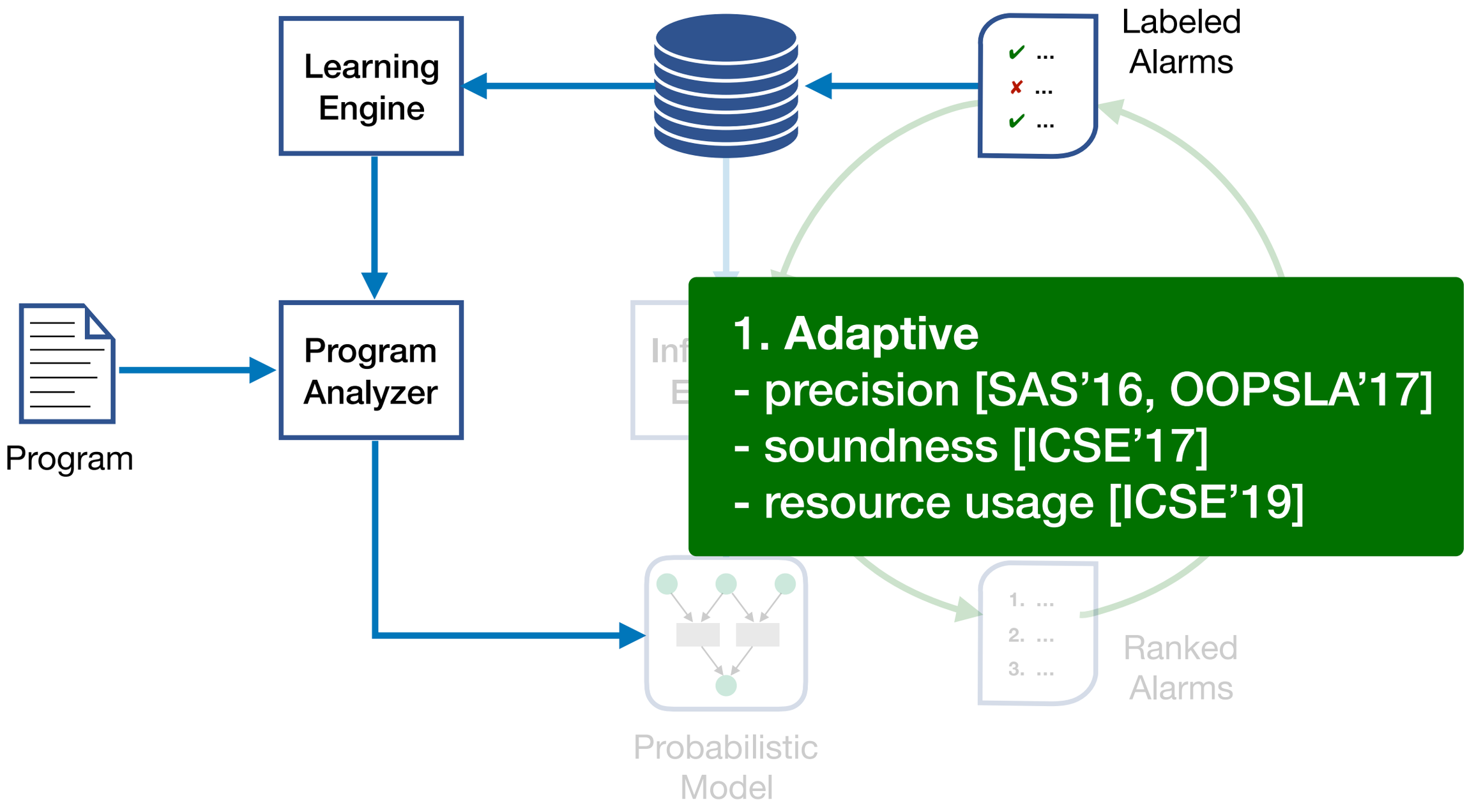


3. Narrow-sighted

Next-generation Program Analysis

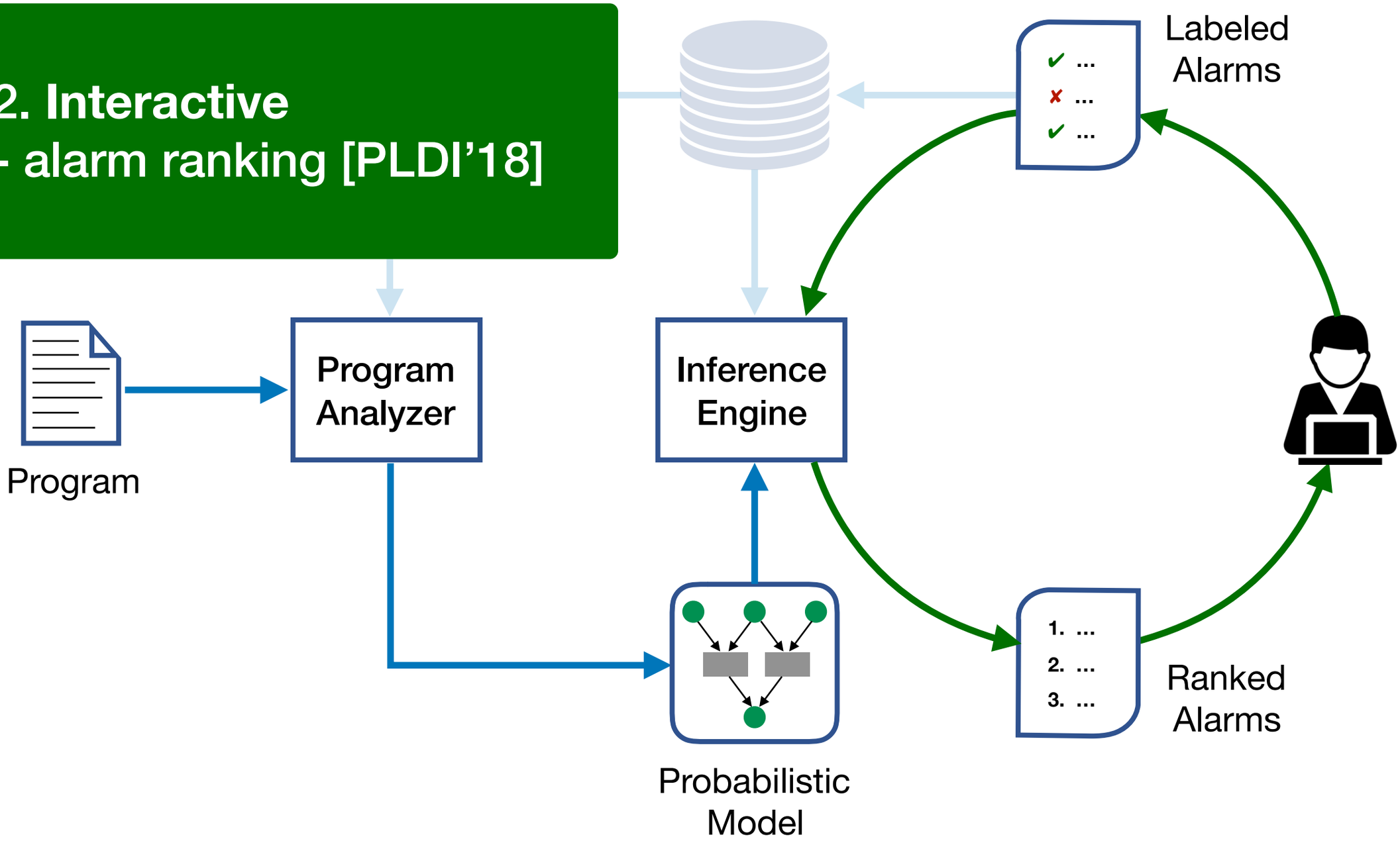


Next-generation Program Analysis



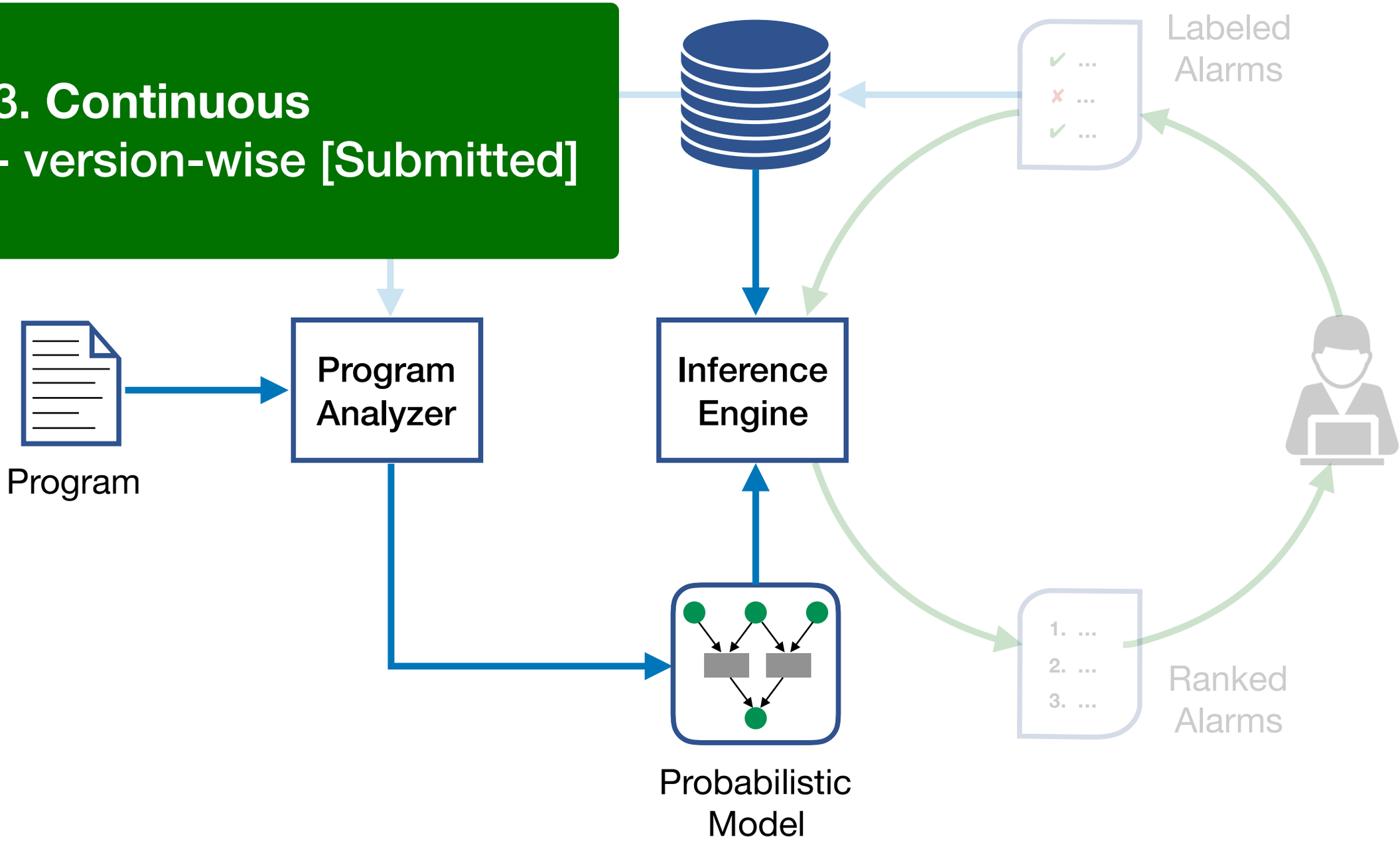
Next-generation Program Analysis

2. Interactive
- alarm ranking [PLDI'18]



Next-generation Program Analysis

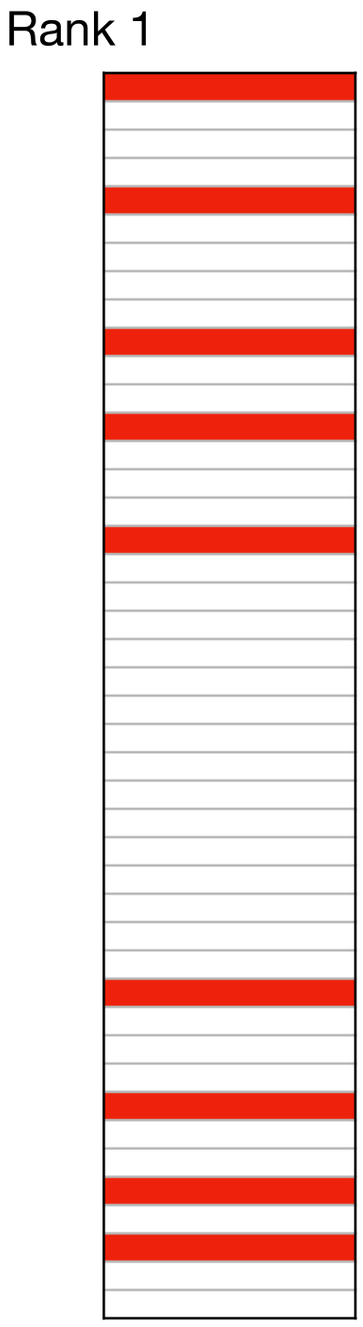
3. Continuous
- version-wise [Submitted]

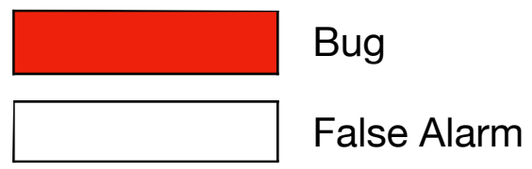


BINGO: An Interactive Alarm Ranking System

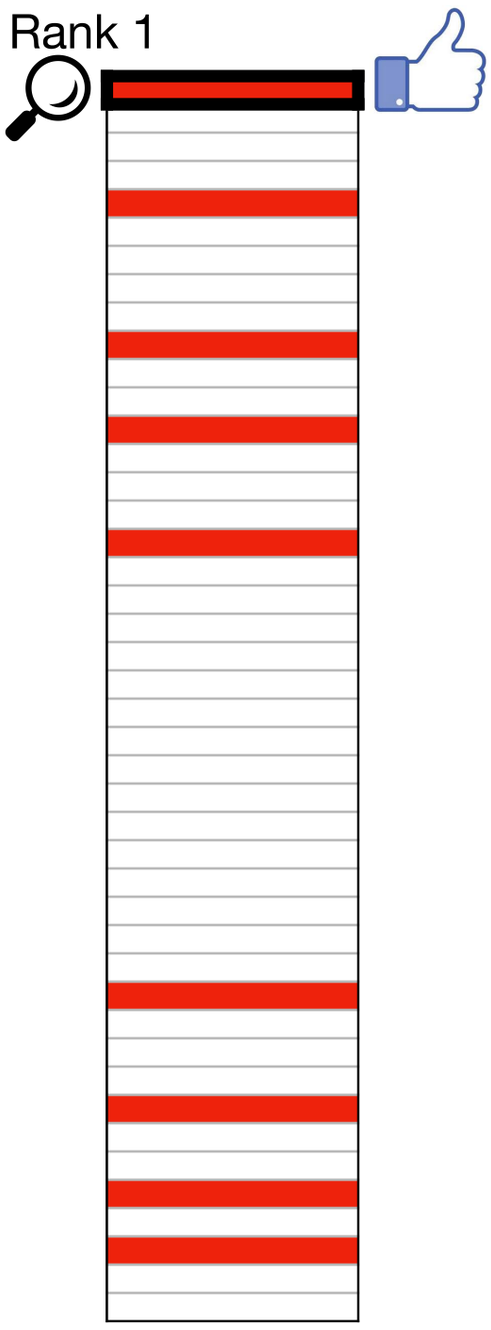
*User-Guided Program Reasoning using Bayesian Inference, PLDI'18

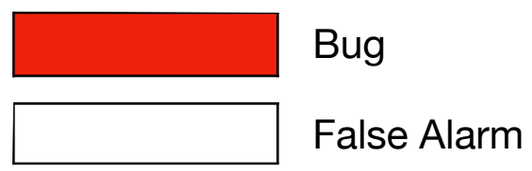
Interactive Alarm Ranker



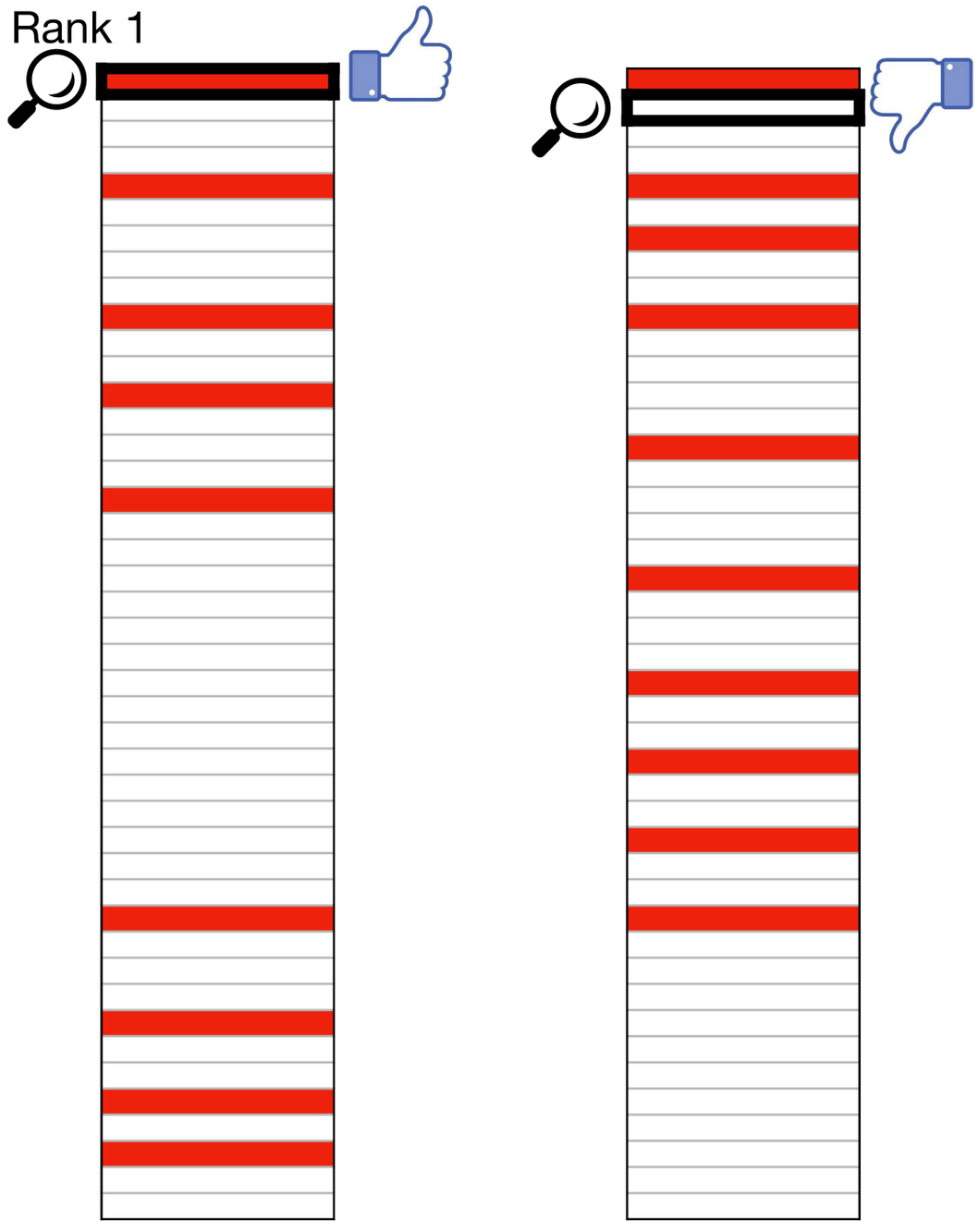


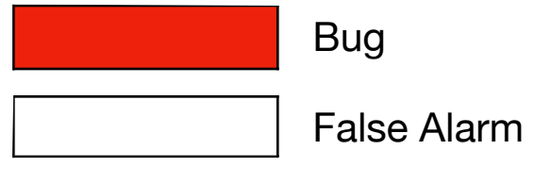
Interactive Alarm Ranker



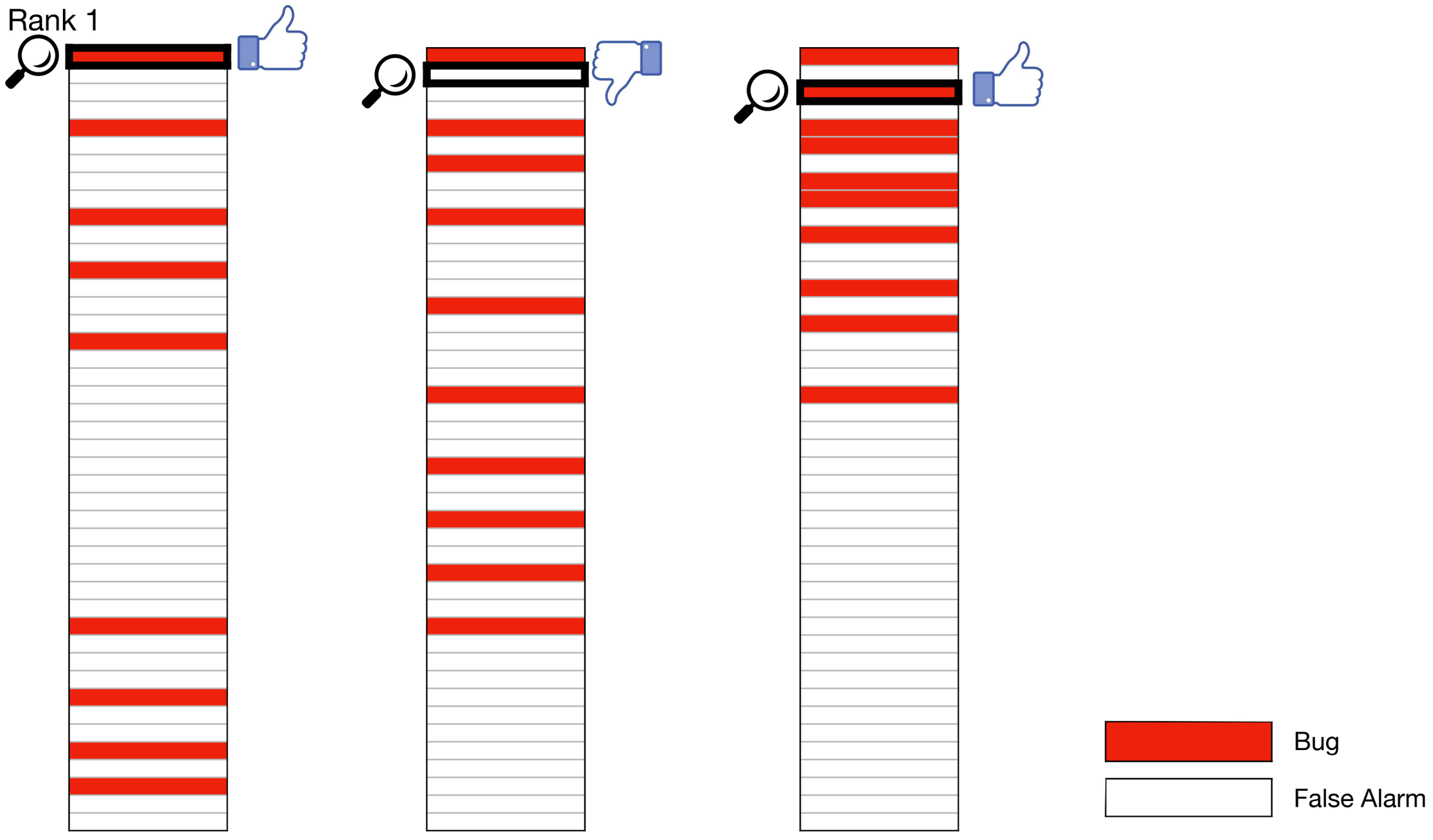


Interactive Alarm Ranker

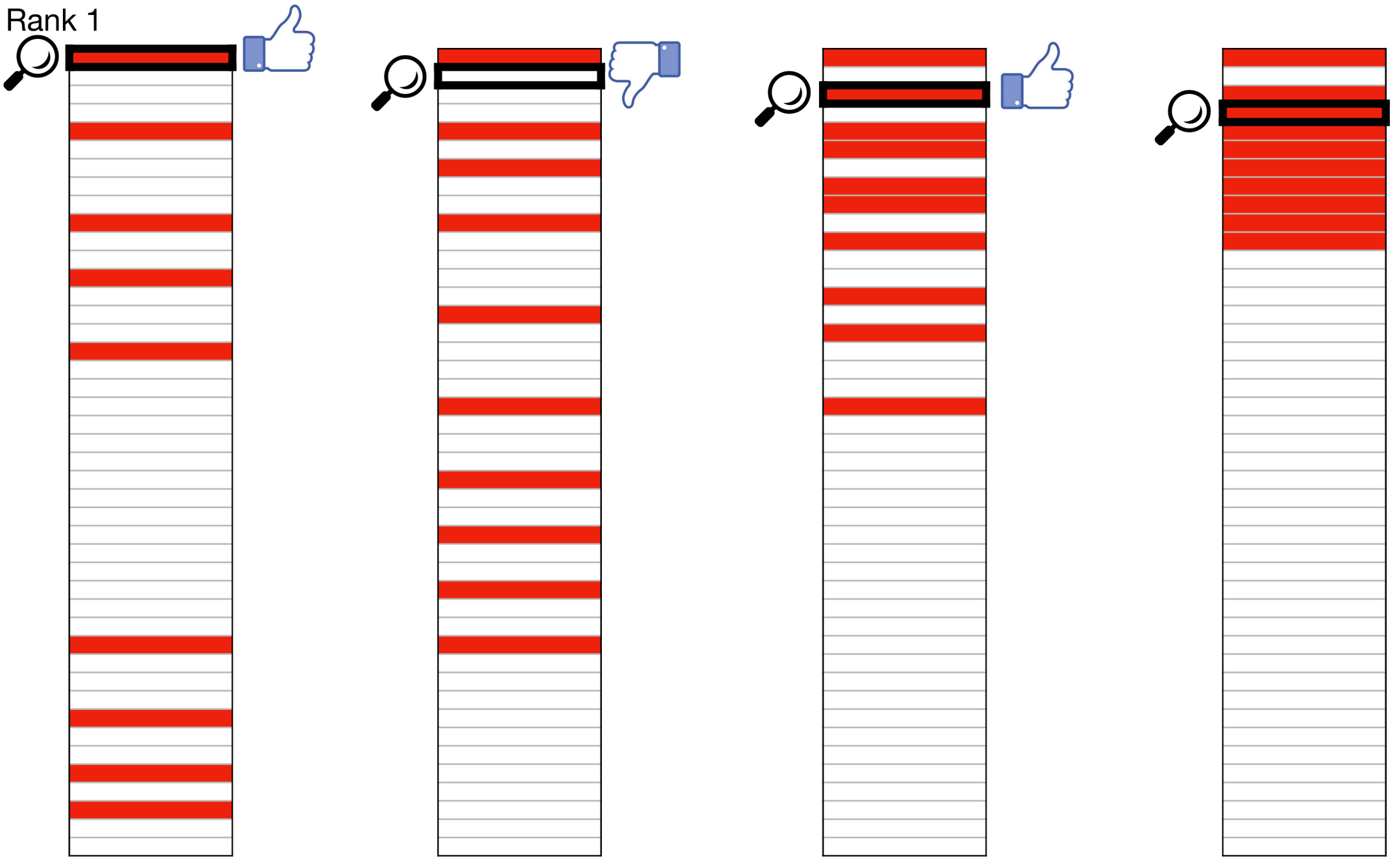




Interactive Alarm Ranker

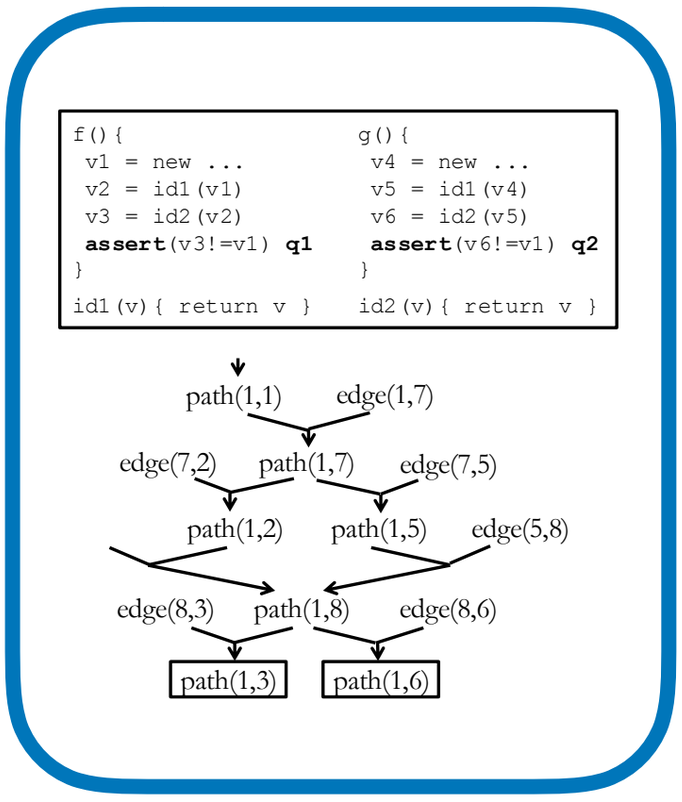


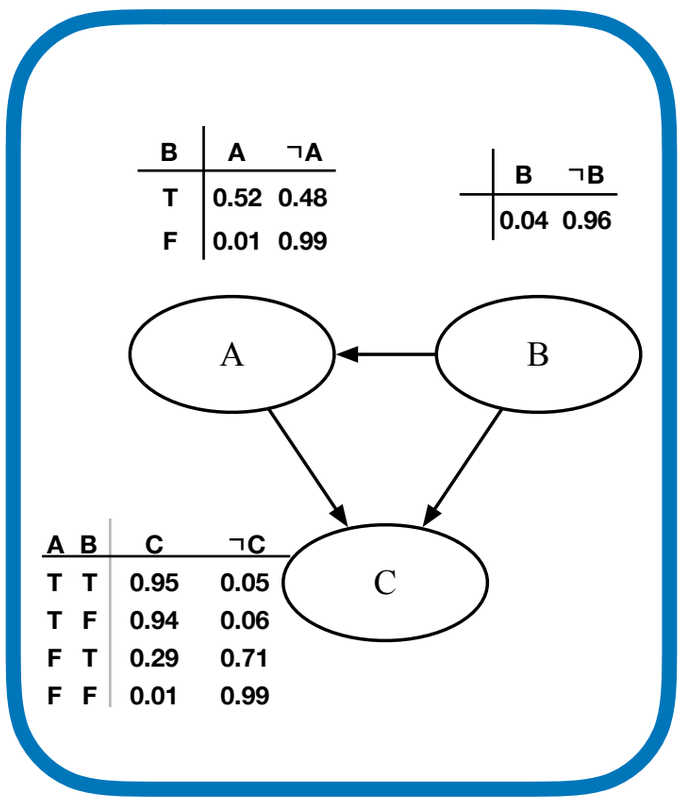
Interactive Alarm Ranker

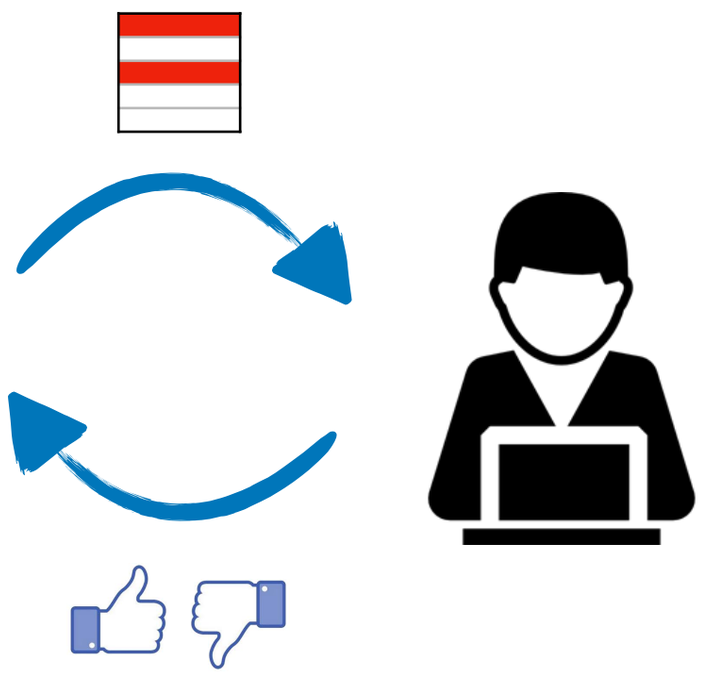


Key Idea

Human in the loop + Bayesian inference





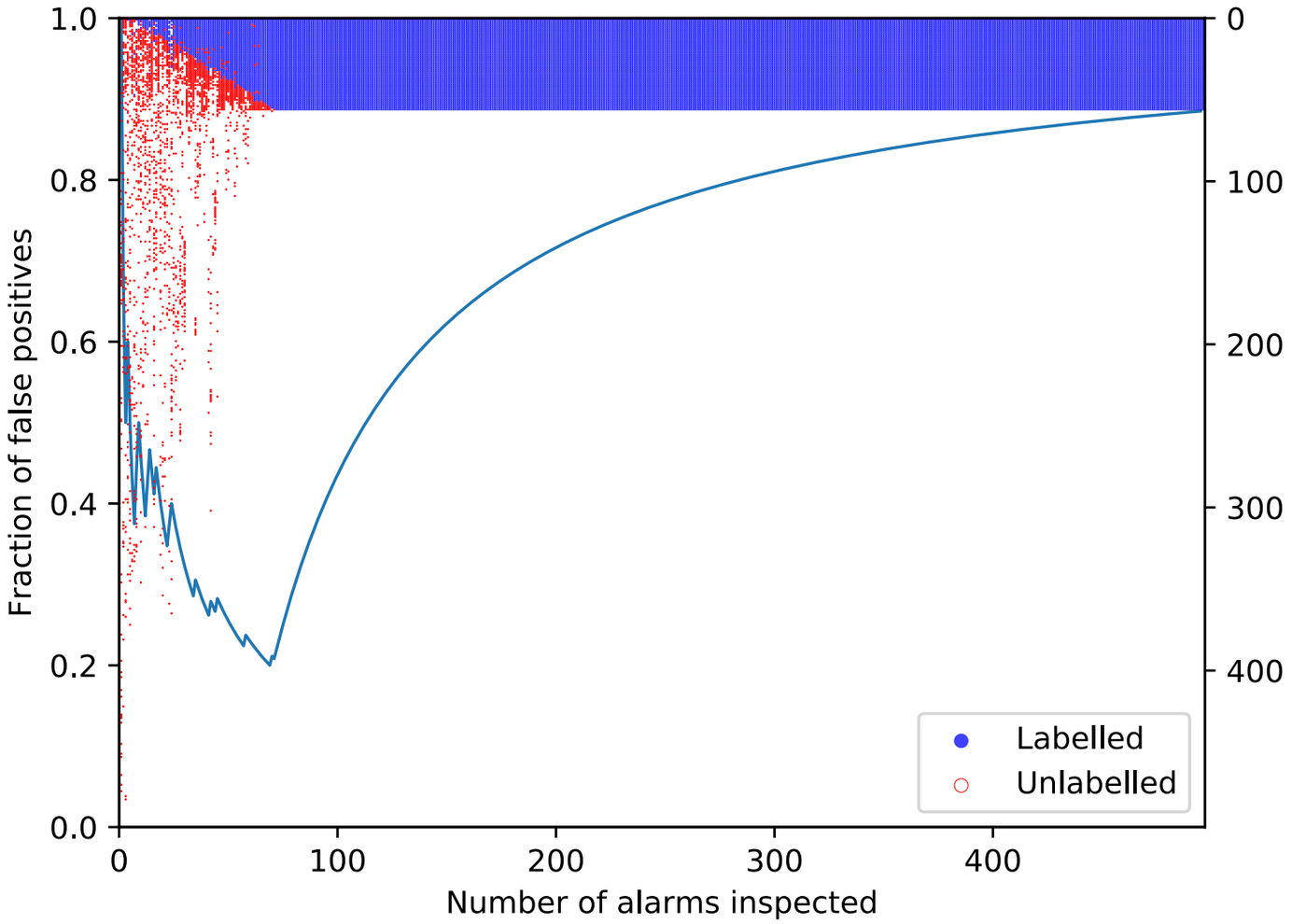


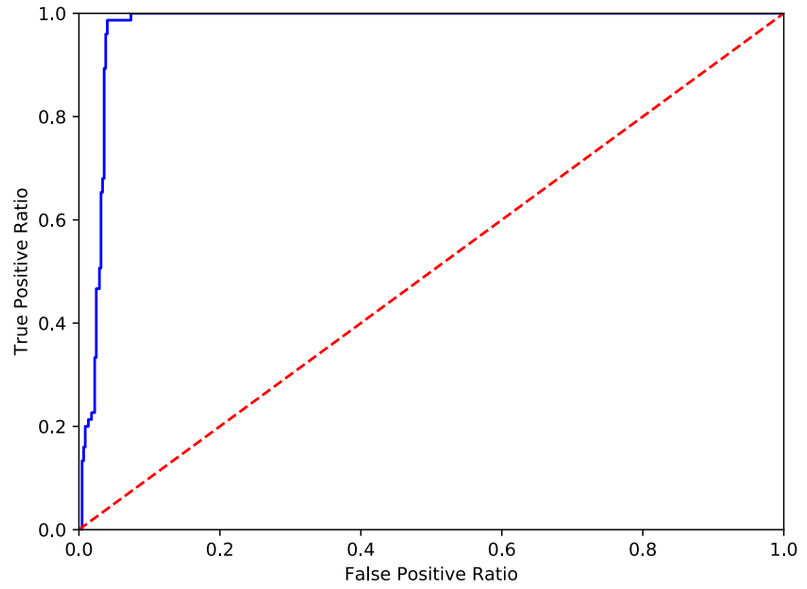
Static Analysis Result

Bayesian Network

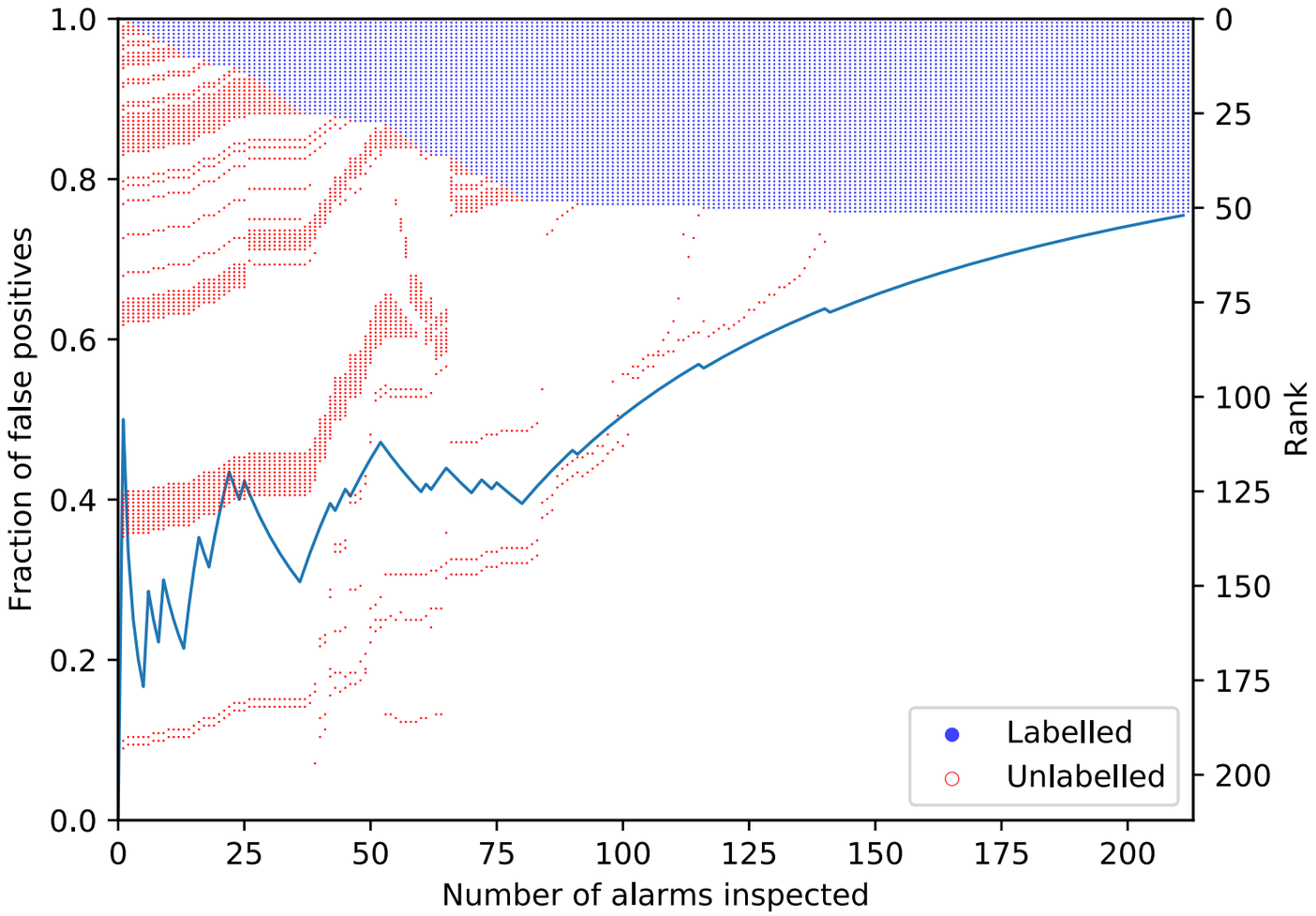
User

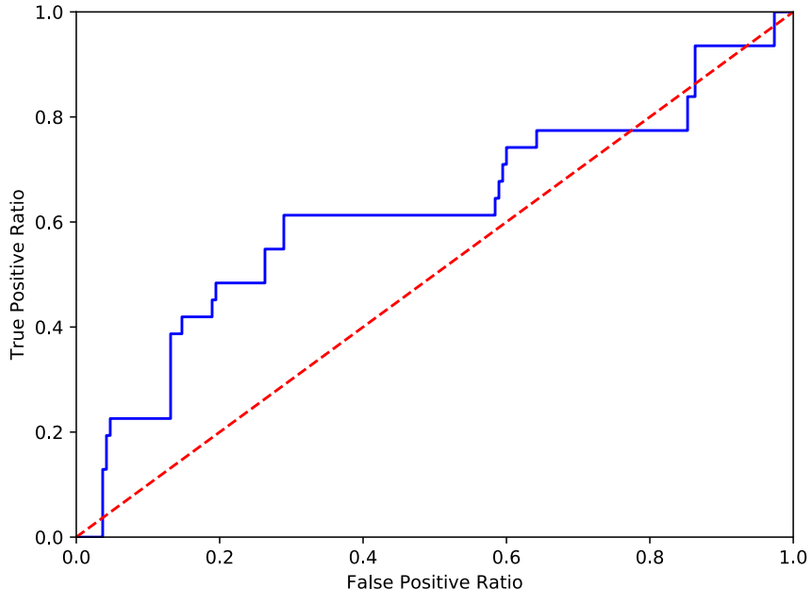
Case Study: Datarace





Case Study: Information Flow





Ex: Datarace Analysis

```
public class RequestHandler {
  private FtpRequest request;

  public FtpRequest getRequest() {
    return request;                    //L0
  }

  public void close() {
    synchronized (this) {              //L1
      if (isClosed) return;            //L2
      isClosed = true;                 //L3
    }
    controlSocket.close();             //L4
    controlSocket = null;              //L5
    request.clear();                   //L6
    request = null;                    //L7
  }
}
```

```
Parallel(p1, p3) :- Parallel(p1, p2), Next(p2, p3),
                    Unguarded(p1, p3).
Parallel(p1, p2) :- Parallel(p2, p1).
  Race(p1, p2) :- Parallel(p1, p2), Alias(p1, p2).
```

***Apache FTP Server**

Ex: Datarace Analysis

```
public class RequestHandler {
  private FtpRequest request;

  public FtpRequest getRequest() {
    return request;                    //L0
  }

  public void close() {
    synchronized (this) {              //L1
      if (isClosed) return;            //L2
      isClosed = true;                 //L3
    }
    controlSocket.close();             //L4
    controlSocket = null;              //L5
    request.clear();                   //L6
    request = null;                    //L7
  }
}
```

```
Parallel(p1, p3) :- Parallel(p1, p2), Next(p2, p3),
                    Unguarded(p1, p3).
Parallel(p1, p2) :- Parallel(p2, p1).
  Race(p1, p2) :- Parallel(p1, p2), Alias(p1, p2).
```

Datarace

***Apache FTP Server**

Ex: Datarace Analysis

```
public class RequestHandler {
  private FtpRequest request;

  public FtpRequest getRequest() {
    return request;                    //L0
  }

  public void close() {
    synchronized (this) {              //L1
      if (isClosed) return;            //L2
      isClosed = true;                 //L3
    }
    controlSocket.close();             //L4
    controlSocket = null;              //L5
    request.clear();                   //L6
    request = null;                    //L7
  }
}
```

False alarm

False alarm

```
Parallel(p1, p3) :- Parallel(p1, p2), Next(p2, p3),
                    Unguarded(p1, p3).
Parallel(p1, p2) :- Parallel(p2, p1).
  Race(p1, p2) :- Parallel(p1, p2), Alias(p1, p2).
```

***Apache FTP Server**

Derivation Graph

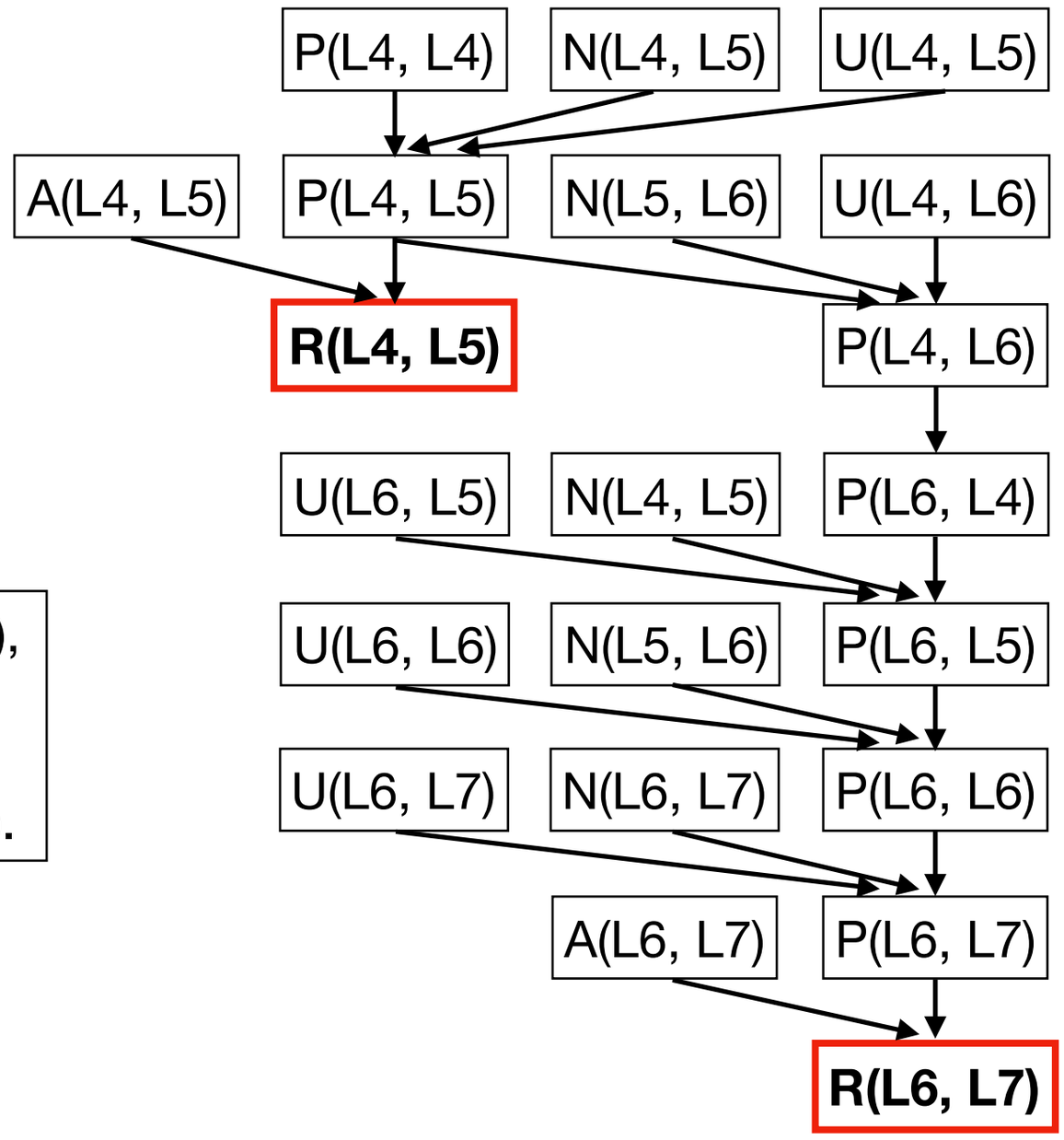
Program

```
controlSocket.close();//L4
controlSocket = null; //L5
request.clear();      //L6
request = null;       //L7
```

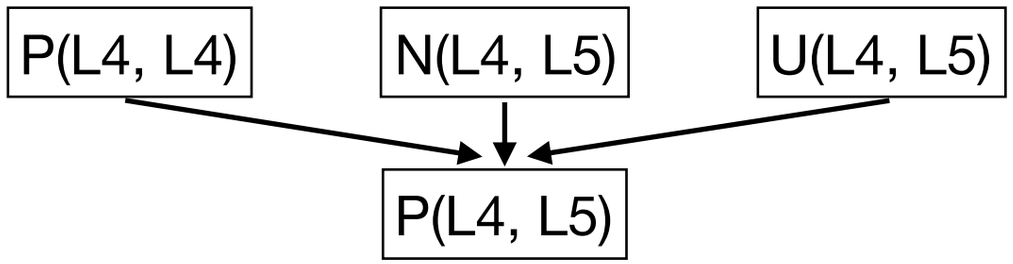
Datalog Rule

```
Parallel(p1, p3) :- Parallel(p1, p2), Next(p2, p3),
                    Unguarded(p1, p3).
Parallel(p1, p2) :- Parallel(p2, p1).
  Race(p1, p2) :- Parallel(p1, p2), Alias(p1, p2).
```

Derivation Graph



Bayesian Network



Logical Rule

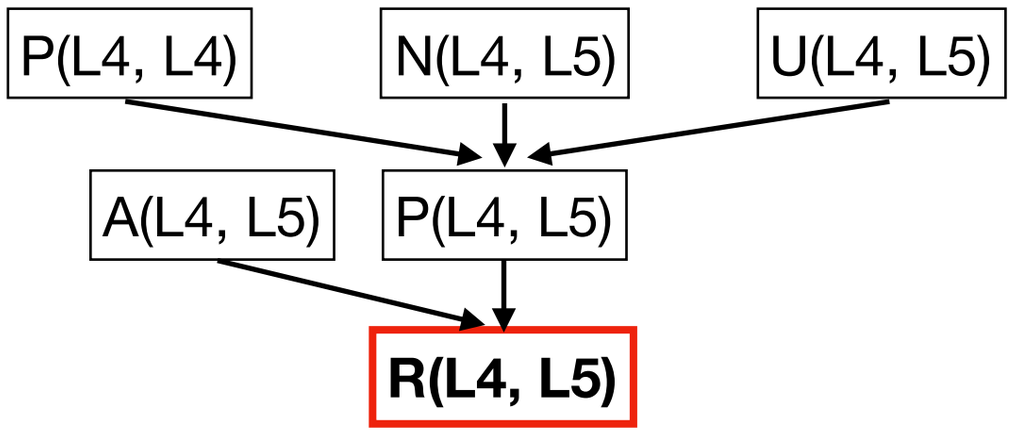
```
Parallel(p1, p3) :- Parallel(p1, p2), Next(p2, p3),
                    Unguarded(p1, p3).
Parallel(p1, p2) :- Parallel(p2, p1).
  Race(p1, p2) :- Parallel(p1, p2), Alias(p1, p2).
```

Probabilistic Rule

P(L4,L4)	N(L4,L5)	U(L4,L5)	Pr(P(L4,L5) | H)
TRUE	TRUE	TRUE	0.95*
TRUE	TRUE	FALSE	0
	…		
FALSE	FALSE	FALSE	0

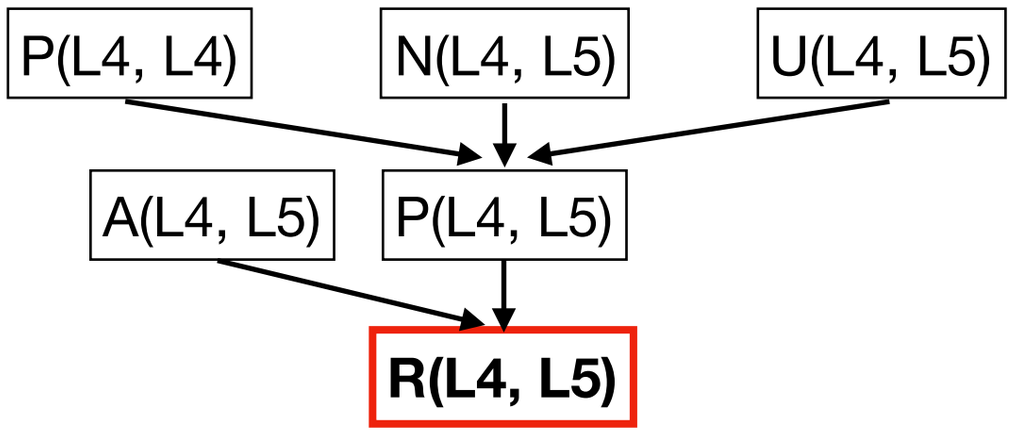
*Prior probability is computed by an offline learning
**Probabilities of input tuples are 1.0

Probability of Alarms



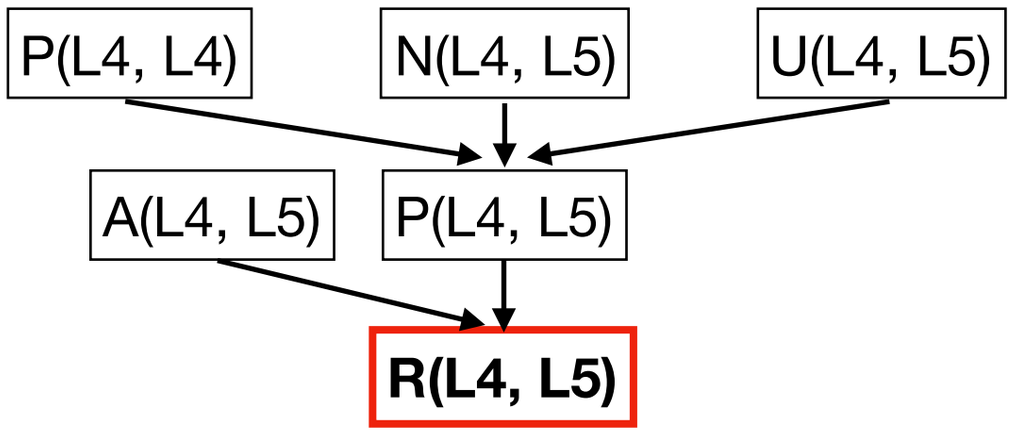
Pr(R(L4,L5)) =

Probability of Alarms



$$\begin{aligned}\Pr(R(L4,L5)) = \Pr(&R(L4,L5), A(L4,L5), P(L4,L5)) \\ &+ \Pr(R(L4,L5), \neg A(L4,L5), P(L4,L5)) \\ &+ \Pr(R(L4,L5), A(L4,L5), \neg P(L4,L5)) \\ &+ \Pr(R(L4,L5), \neg A(L4,L5), \neg P(L4,L5))\end{aligned}$$

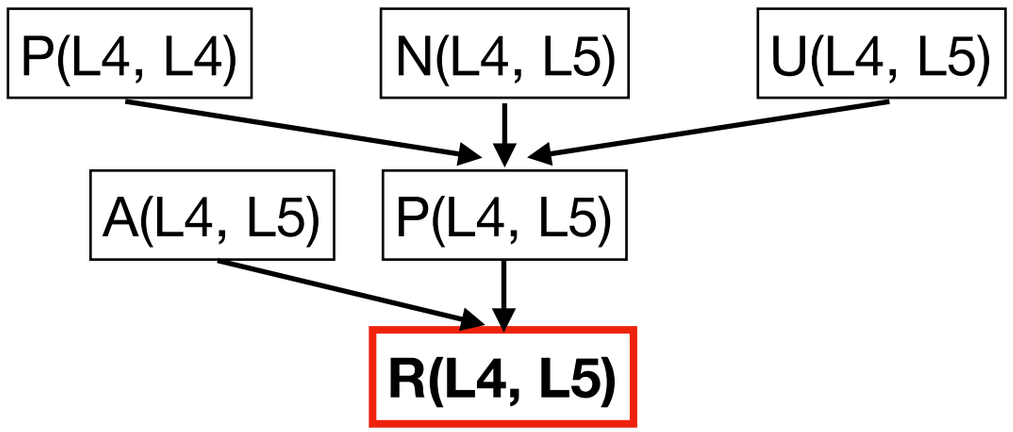
Probability of Alarms



$$\Pr(R(L4,L5)) = \Pr(R(L4,L5), A(L4,L5), P(L4,L5))$$
$$+ \Pr(R(L4,L5), \neg A(L4,L5), P(L4,L5))$$
$$+ \Pr(R(L4,L5), A(L4,L5), \neg P(L4,L5))$$
$$+ \Pr(R(L4,L5), \neg A(L4,L5), \neg P(L4,L5))$$

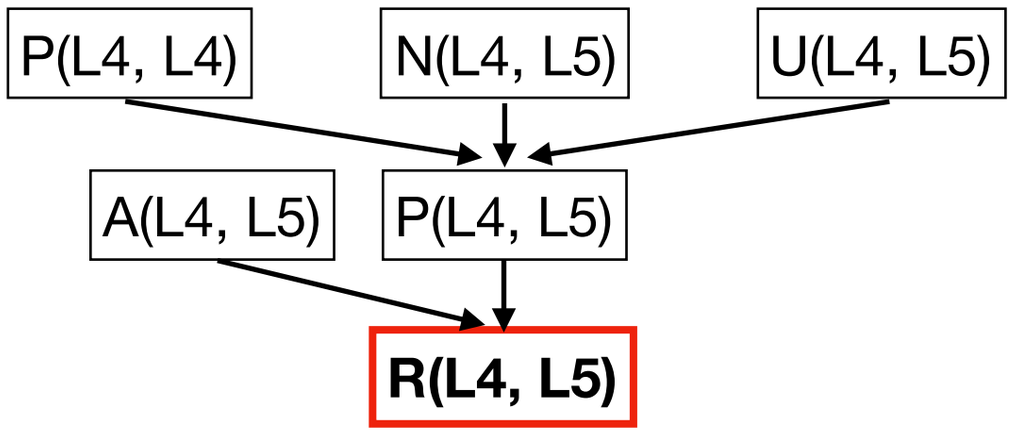
If any of the antecedents fail, then the race cannot happen.

Probability of Alarms



Pr(R(L4,L5)) = Pr(R(L4,L5), A(L4,L5), P(L4,L5))

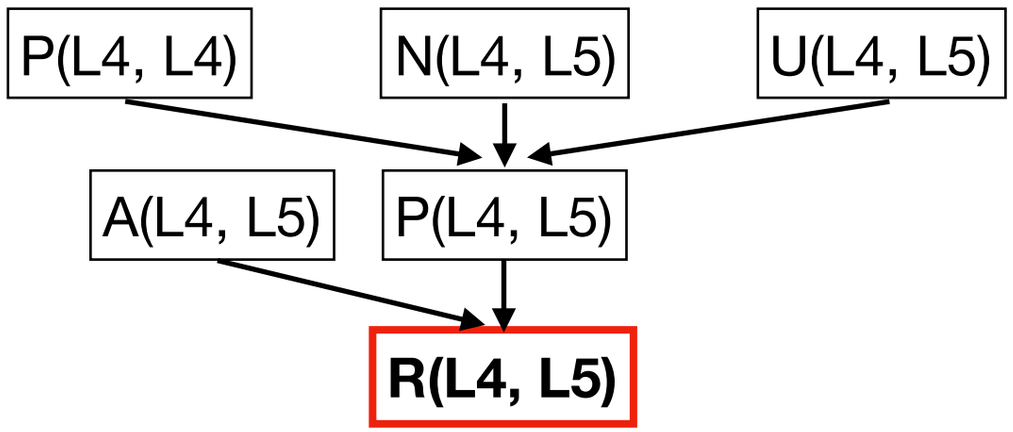
Probability of Alarms



$$\Pr(R(L4,L5)) = \Pr(R(L4,L5), A(L4,L5), P(L4,L5))$$
$$= \Pr(R(L4,L5) \mid A(L4,L5), P(L4,L5)) * \Pr(A(L4,L5)) * \Pr(P(L4,L5))$$

By Bayes's Rule:
$\Pr(A,B) = \Pr(A|B) * \Pr(B)$

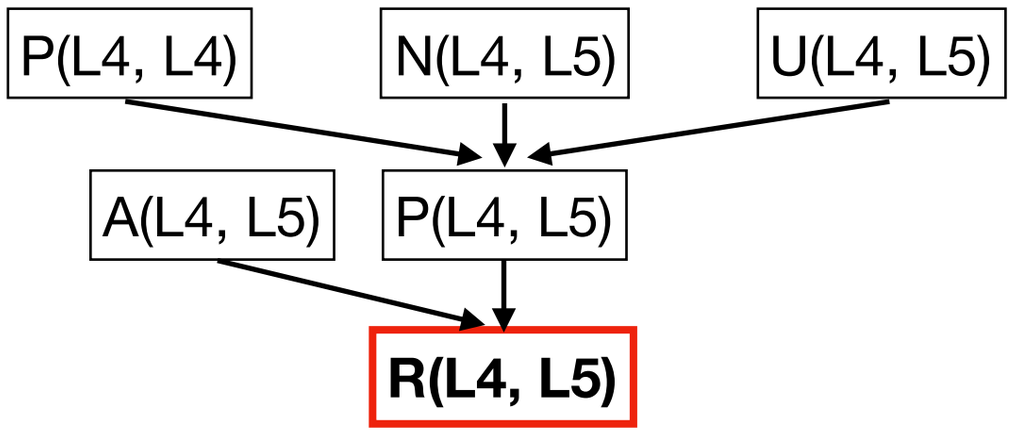
Probability of Alarms



$$
\begin{aligned}
Pr(R(L4,L5)) &= Pr(R(L4,L5), A(L4,L5), P(L4,L5)) \\
&= Pr(R(L4,L5) \mid A(L4,L5), P(L4,L5)) * \\
&\quad Pr(A(L4,L5)) * Pr(P(L4,L5)) \\
&= 0.95 * 1.0 * Pr(P(L4,L5)) \\
&= 0.95 * Pr(P(L4,L5), Pr(P(L4,L4)), Pr(N(L4,L5), Pr(U(L4,L5))
\end{aligned}
$$

Assume that the probabilities of firing each rule and input tuple are 0.95 and 1.0.

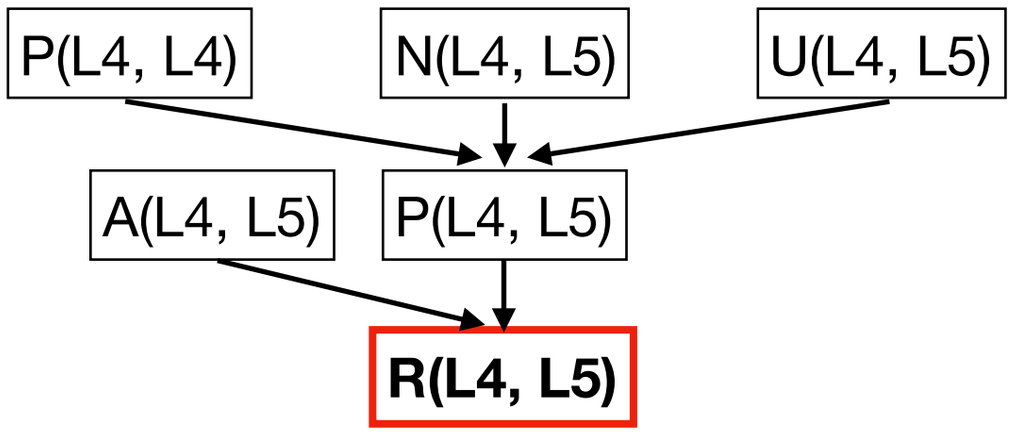
Probability of Alarms



$$
\begin{aligned}
Pr(R(L4,L5)) &= Pr(R(L4,L5), A(L4,L5), P(L4,L5)) \\
&= Pr(R(L4,L5) \mid A(L4,L5), P(L4,L5)) * \\
&\quad Pr(A(L4,L5)) * Pr(P(L4,L5)) \\
&= 0.95 * 1.0 * Pr(P(L4,L5)) \\
&= 0.95 * Pr(P(L4,L5), Pr(P(L4,L4)), Pr(N(L4,L5), Pr(U(L4,L5)) \\
&= 0.95 * Pr(P(L4,L5) \mid Pr(P(L4,L4)), Pr(N(L4,L5), Pr(U(L4,L5)) * \\
&\quad Pr(P(L4,L4)) * Pr(N(L4,L5)) * Pr(U(L4,L5))
\end{aligned}
$$

By Bayes's Rule:
$Pr(A,B) = Pr(A|B) * Pr(B)$

Probability of Alarms



$$
\begin{aligned}
Pr(R(L4,L5)) &= Pr(R(L4,L5), A(L4,L5), P(L4,L5)) \\
&= Pr(R(L4,L5) \mid A(L4,L5), P(L4,L5)) * \\
&\quad Pr(A(L4,L5)) * Pr(P(L4,L5)) \\
&= 0.95 * 1.0 * Pr(P(L4,L5)) \\
&= 0.95 * 0.95 * Pr(P(L4,L4)) * Pr(N(L4,L5) * Pr(U(L4,L5)) \\
&= \ldots \\
&= 0.398
\end{aligned}
$$

Alarm Ranking

```
public class RequestHandler {
  private FtpRequest request;

  public FtpRequest getRequest() {
    return request;                    //L0
  }


  public void close() {
    synchronized (this) {              //L1
      if (isClosed) return;            //L2
      isClosed = true;                 //L3
    }
    controlSocket.close();             //L4
    controlSocket = null;              //L5
    request.clear();                   //L6
    request = null;                    //L7
  }
}
```

Ranking	Alarm	Confidence
1	R(L4, L5)	0.398
2	R(L5, L5)	0.378
3	R(L6, L7)	0.324
4	R(L7, L7)	0.308
5	R(L0, L7)	0.279

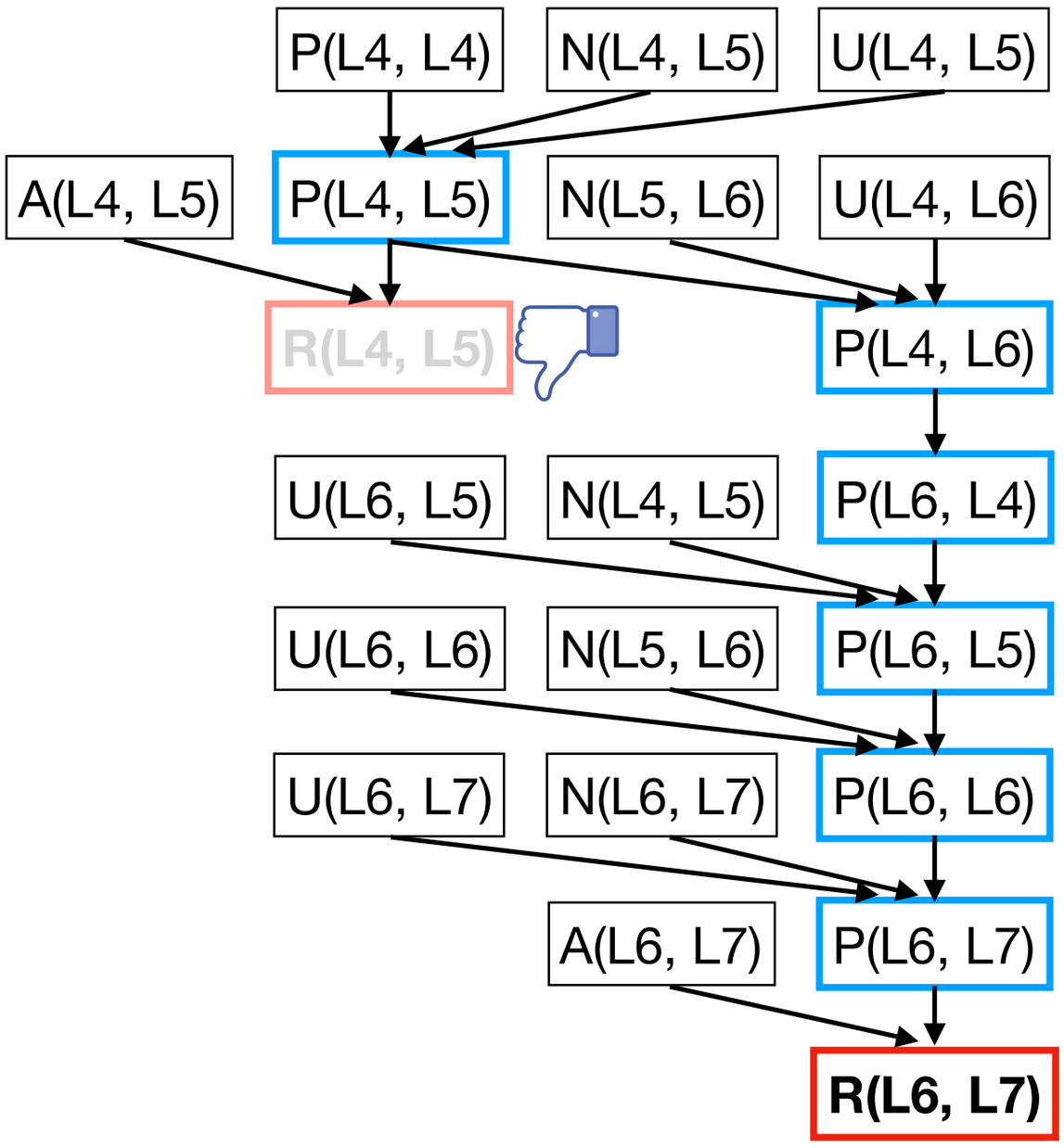
Alarm Ranking

```
public class RequestHandler {
  private FtpRequest request;

  public FtpRequest getRequest() {
    return request;                    //L0
  }

  public void close() {
    synchronized (this) {              //L1
      if (isClosed) return;            //L2
      isClosed = true;                 //L3
    }
    controlSocket.close();             //L4
    controlSocket = null;              //L5
    request.clear();                   //L6
    request = null;                    //L7
  }
}
```

Ranking	Alarm	Confidence
1	R(L4, L5)	0.398
2	R(L5, L5)	0.378
3	R(L6, L7)	0.324
4	R(L7, L7)	0.308
5	R(L0, L7)	0.279

Q: What are the probabilities of the other alarms when R(L4,L5) is false?

Marginal Inference



Pr(P(L4,L5) | ¬R(L4,L5))
= Pr(¬R(L4,L5) | P(L4,L5)) *
Pr(P(L4,L5)) / Pr(¬R(L4,L5))
= 0.03

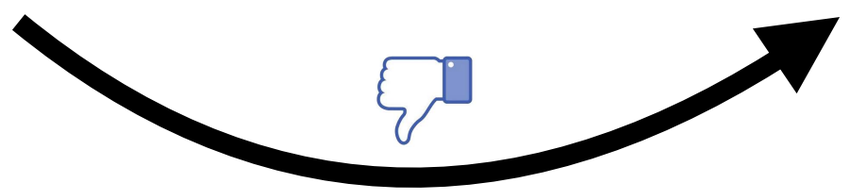
By Bayes's Rule:
Pr(A|B) = P(B|A) * Pr(A) / Pr(B)

Pr(R(L6,L7) | ¬R(L4,L5))
= Pr(R(L6,L7) | P(L4,L5)) *
Pr(P(L4,L5)) | ¬R(L4,L5))
= 0.03

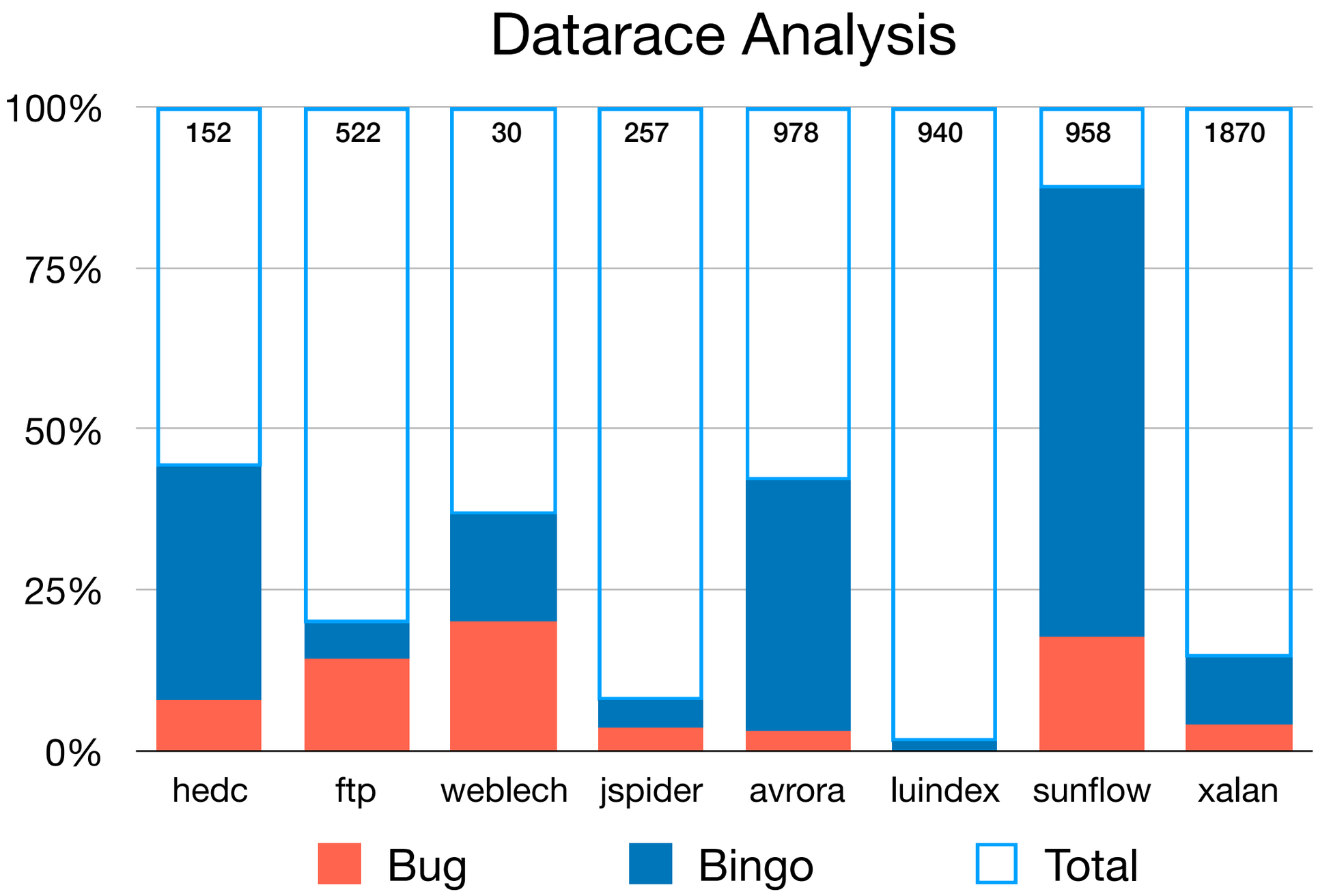
Alarm Ranking

Ranking	Alarm	Confidence
1	R(L4, L5)	0.398
2	R(L5, L5)	0.378
3	R(L6, L7)	0.324
4	R(L7, L7)	0.308
5	R(L0, L7)	0.279

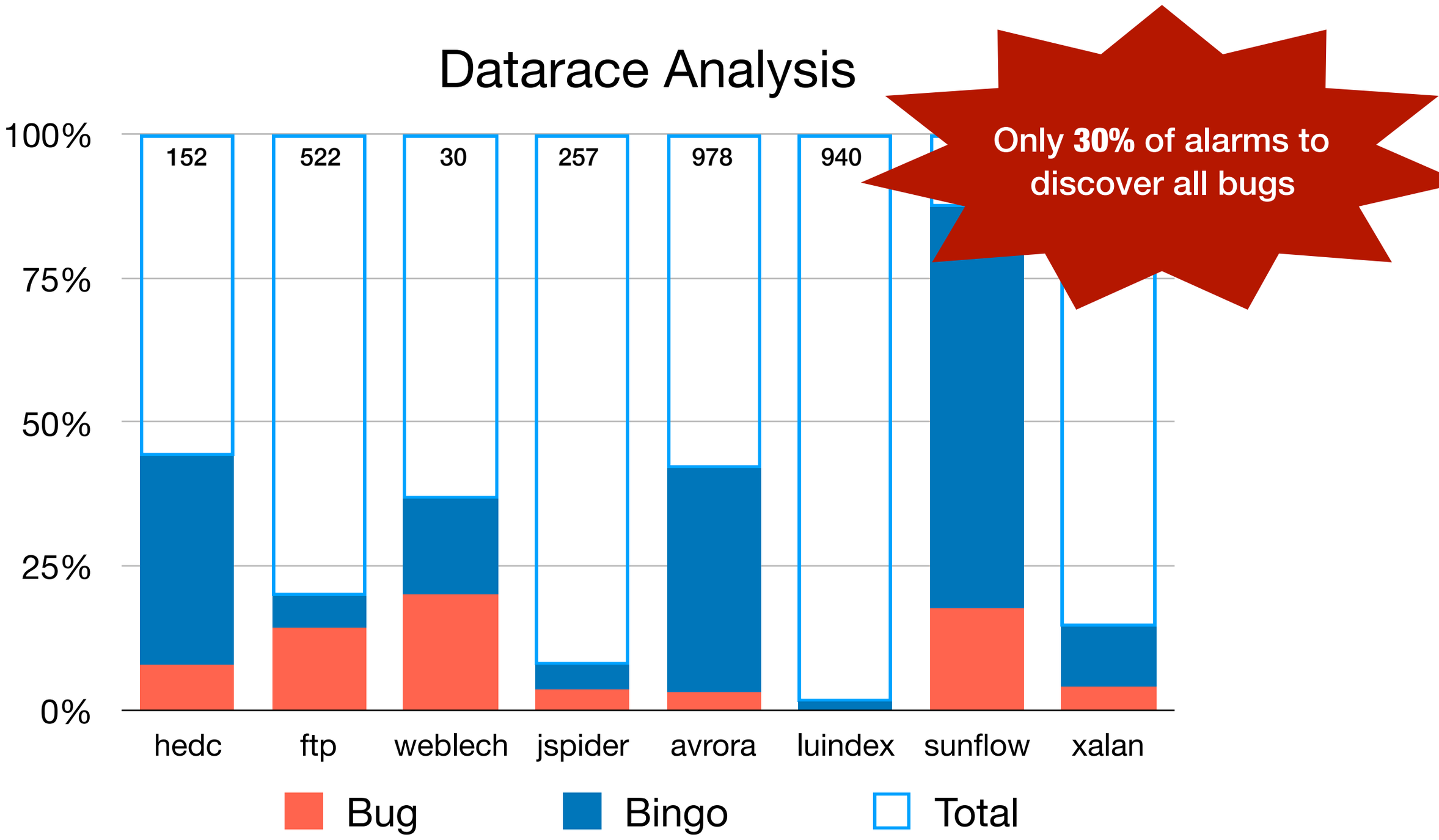
Ranking	Alarm	Confidence
1	R(L0, L7)	0.279
2	R(L5, L5)	0.035
3	R(L6, L7)	0.030
4	R(L7, L7)	0.028
5	R(L4, L5)	0



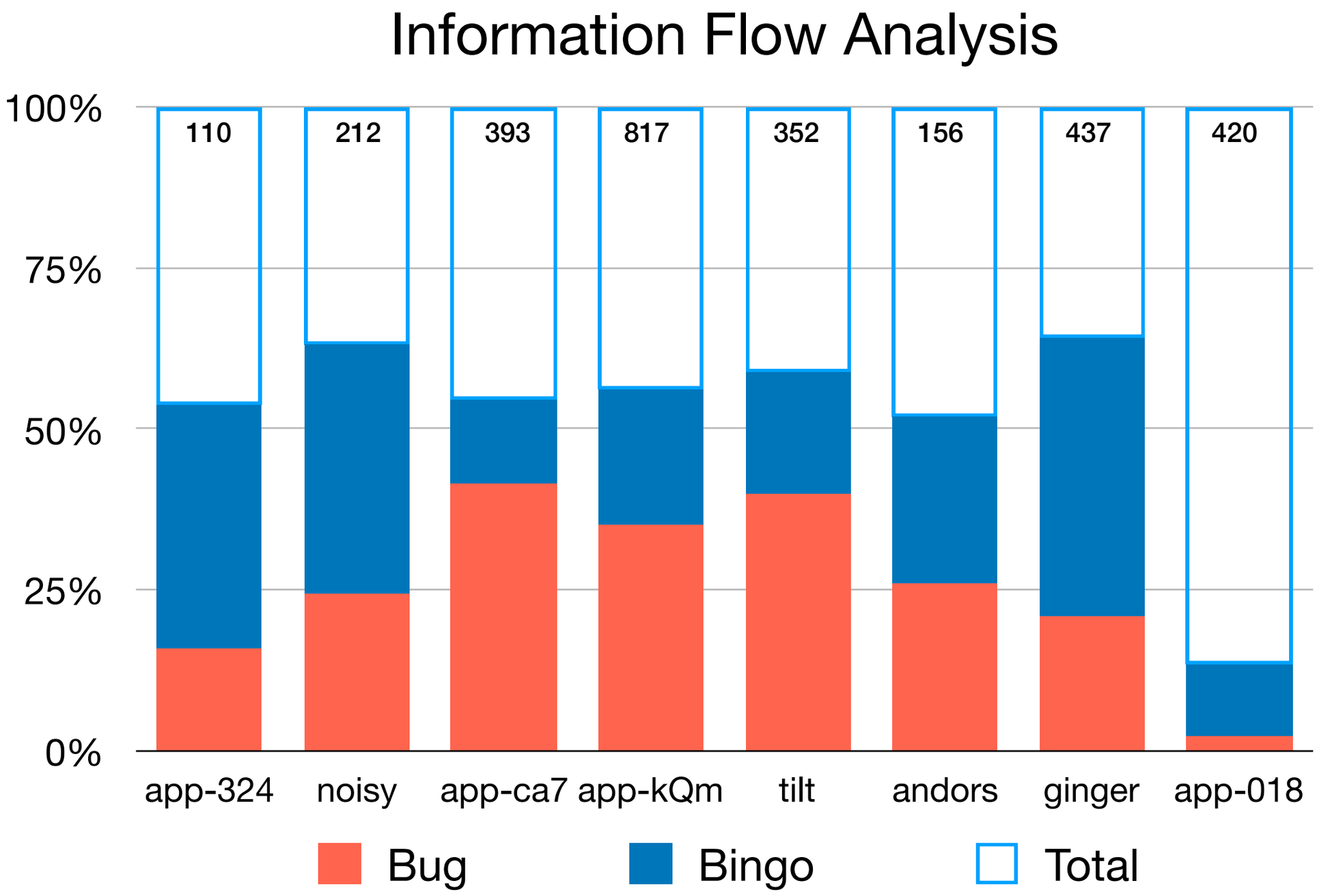
Experimental Results



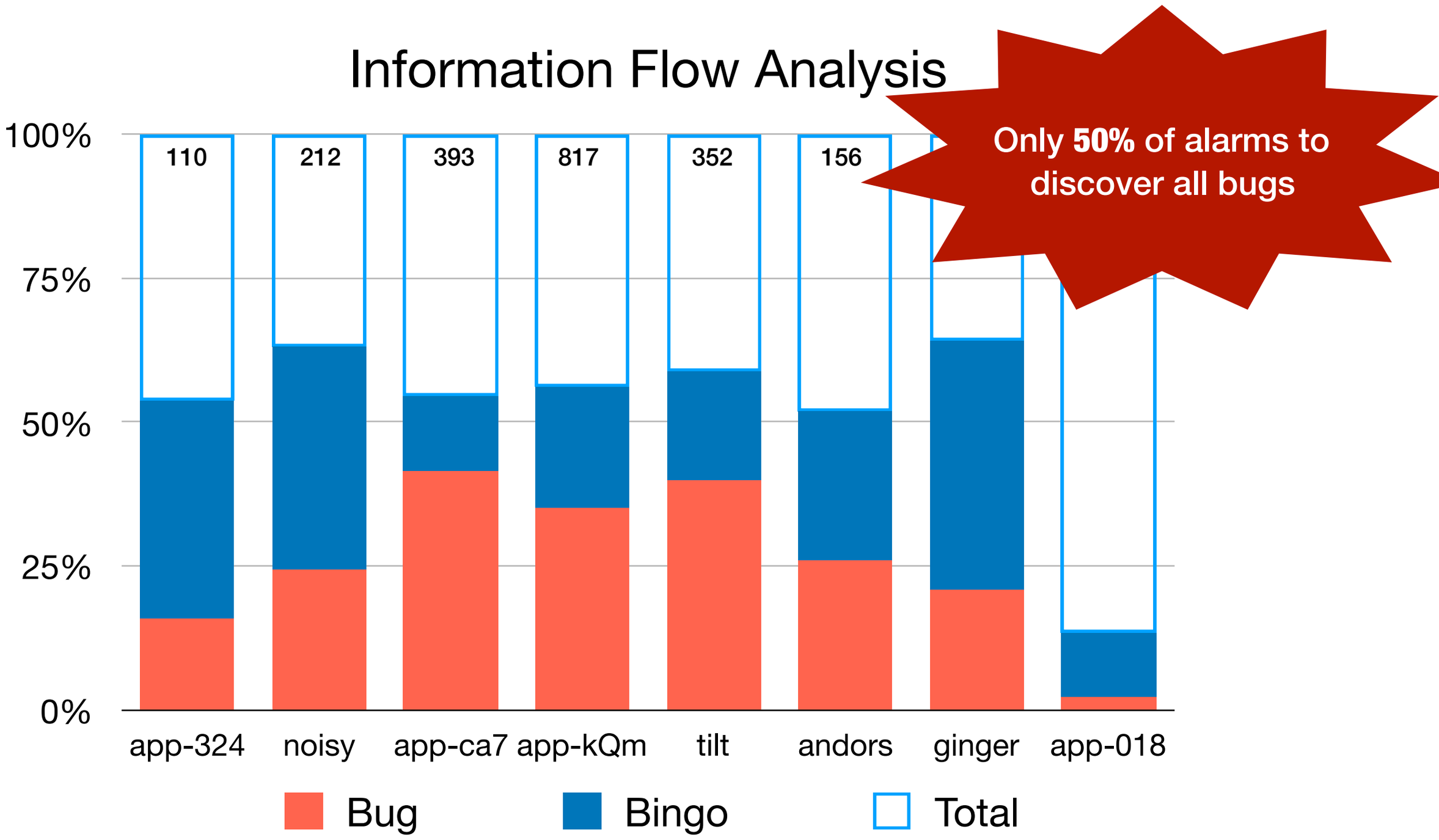
Experimental Results



Experimental Results



Experimental Results



Drake: A Continuous Program Reasoning Framework

*Continuous Program Reasoning via Differential Bayesian Inference, In submission

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}

wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

-  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+         shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}

wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

-  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+         shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}


wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

-  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+         shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}


wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+         shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}

wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

-  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+         shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}

wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

-  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+         shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}


wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

-  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+          shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Example

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}


wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

```
void cmp_main(char *filename1, char *filename2) {
  wave_info *info1, *info2;
  long bytes;
  char *buf;

  info1 = new_wave_info(filename1);
  info2 = new_wave_info(filename2);

-  bytes = min(min(info1->data_size, info2->data_size), CMP_SIZE);
+  cmp_size = shift_secs * info1->rate;  // Integer Overflow
+  bytes = min(min(info1->data_size, info2->data_size), cmp_size);

  buf = malloc(2 * bytes * sizeof(char));  // Alarm 2
  /* compare two wave files */
}

int main(int argc, char *argv) {
  int c ;
  while ((c = getopt(argc, argv, "c:f:ls")) != -1) {
      switch (c) {
      case 'c':
+         shift_secs = atoi(optarg);  // Input Source
          cmp_main(argv[optind], argv[optind + 1]);
          break;
      case 't':
          trim_main(argv[optind]);
          break;
  }
  return 0;
}
```

Program Analysis

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}

wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
  info = new_wave_info(filename);
  long header_size;
  char *header;

  header_size = min(info->header_size, HEADER_SIZE);
  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

Program Analysis

```
- #define CMP_SIZE 529200
  #define HEADER_SIZE 44
+ int shift_secs;

  void read_value_long(FILE *file, long *val) {
    char buf[5];
7:  fread(buf, 1, 4, file); // Input Source
    buf[4] = 0;
9:  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
  }

  wave_info *new_wave_info(char *filename) {
    wave_info *info;
    FILE *f;

    info = malloc(sizeof(wave_info));
    f = fopen(filename);
18: read_value_long(f, info->header_size);
    read_value_long(f, info->data_size);
    return info;
  }

  void trim_main(char *filename) {
    wave_info *info;
25: info = new_wave_info(filename);
    long header_size;
    char *header;

29: header_size = min(info->header_size, HEADER_SIZE);
30: header = malloc(header_size * sizeof(char)); // Alarm 1
    /* trim a wave file */
  }
```

Program Analysis

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
7:  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
9:  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}

wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
18:  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
25:  info = new_wave_info(filename);
  long header_size;
  char *header;

29:  header_size = min(info->header_size, HEADER_SIZE);
30:  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

Input relations

DUEdge(c1, c2) : Immediate data flow c1 to c2
Src(c) : Origin of potentially erroneous traces
Dst(c) : Potential program crash point

Output relations

DUPath(c1, c2) : Transitive data flow from c1 to c2
Alarm(c) : Potentially erroneous trace reaching c

Analysis Rules

r1 : DUPath(c1, c2) :- DUEdge(c1, c2).
r2 : DUPath(c1, c3) :- DUPath(c1, c2), DUEdge(c1, c2).
r3 : Alarm(c2) :- DUPath(c1, c2), Src(c1), Dst(c2).

Program Analysis

```
- #define CMP_SIZE 529200
#define HEADER_SIZE 44
+ int shift_secs;

void read_value_long(FILE *file, long *val) {
  char buf[5];
7:  fread(buf, 1, 4, file); // Input Source
  buf[4] = 0;
9:  *val = (buf[3]<<24)|(buf[2]<<16)|(buf[1]<<8)|buf[0];
}

wave_info *new_wave_info(char *filename) {
  wave_info *info;
  FILE *f;

  info = malloc(sizeof(wave_info));
  f = fopen(filename);
18:  read_value_long(f, info->header_size);
  read_value_long(f, info->data_size);
  return info;
}

void trim_main(char *filename) {
  wave_info *info;
25:  info = new_wave_info(filename);
  long header_size;
  char *header;

29:  header_size = min(info->header_size, HEADER_SIZE);
30:  header = malloc(header_size * sizeof(char)); // Alarm 1
  /* trim a wave file */
}
```

Input relations

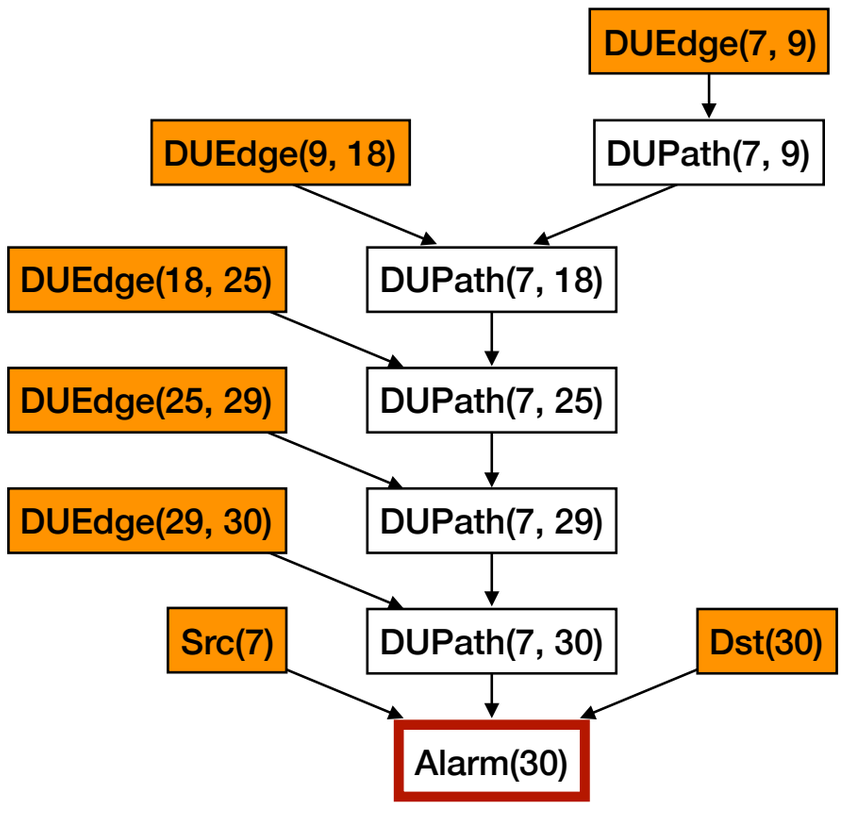
DUEdge(c1, c2) : Immediate data flow c1 to c2
Src(c) : Origin of potentially erroneous traces
Dst(c) : Potential program crash point

Output relations

DUPath(c1, c2) : Transitive data flow from c1 to c2
Alarm(c) : Potentially erroneous trace reaching c

Analysis Rules

r1 : DUPath(c1, c2) :- DUEdge(c1, c2).
r2 : DUPath(c1, c3) :- DUPath(c1, c2), DUEdge(c1, c2).
r3 : Alarm(c2) :- DUPath(c1, c2), Src(c1), Dst(c2).



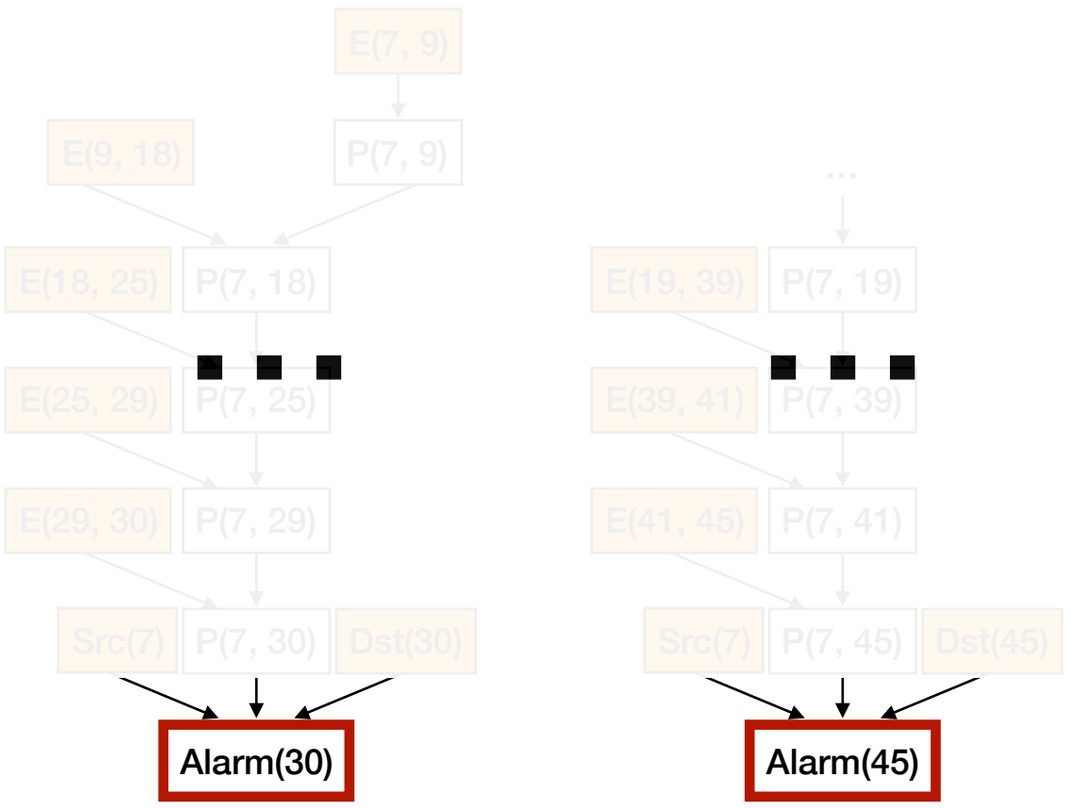
Differential Reasoning

Analysis Results of the Old Version

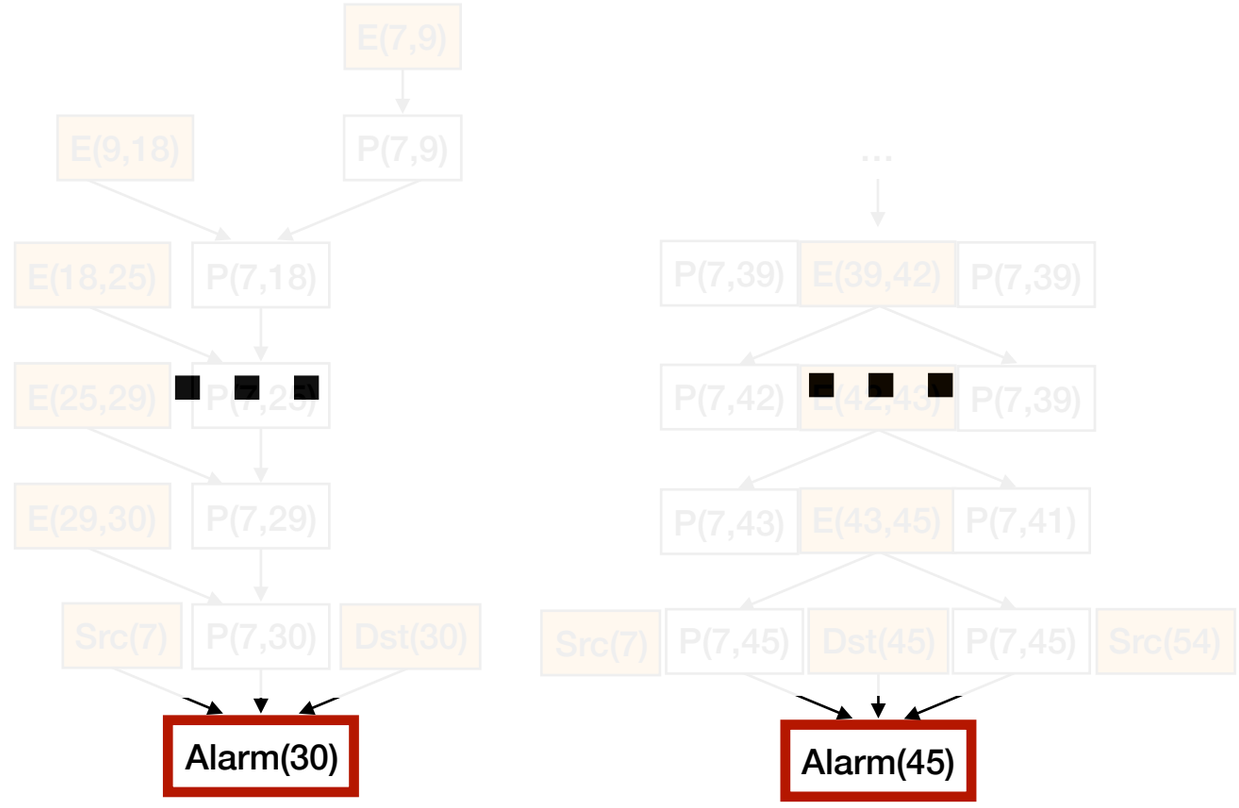
Analysis Results of the New Version

Differential Reasoning

Analysis Results of the **Old** Version

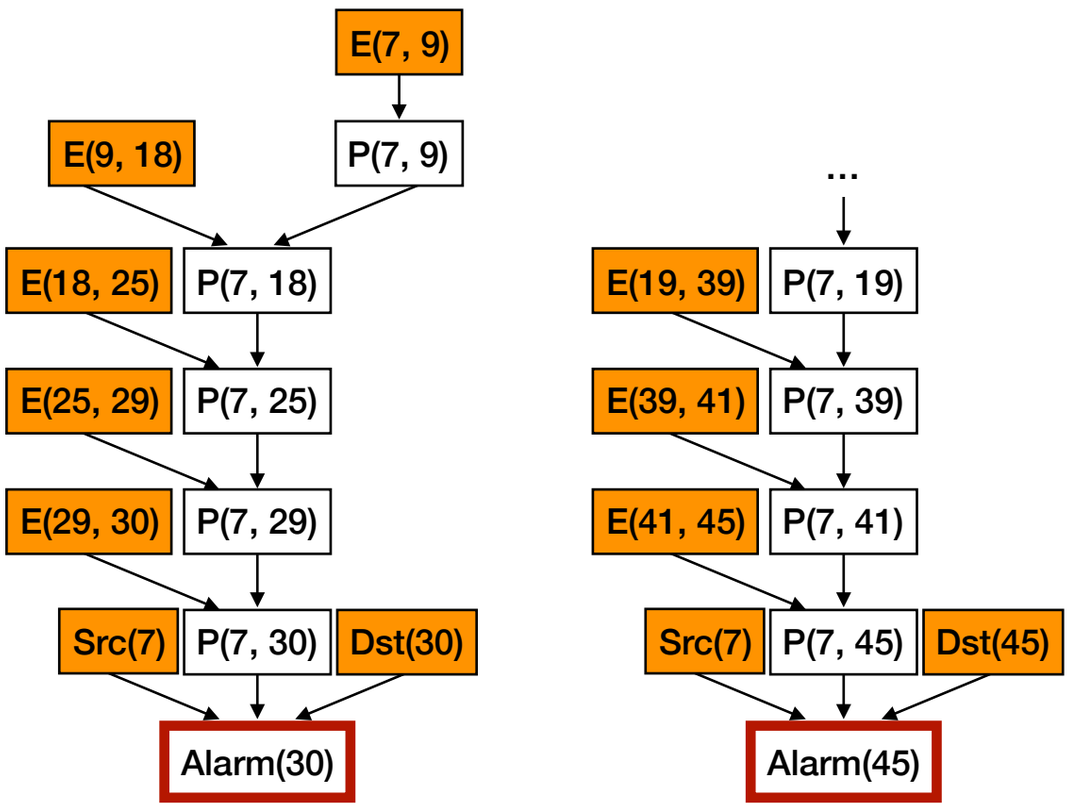


Analysis Results of the **New** Version

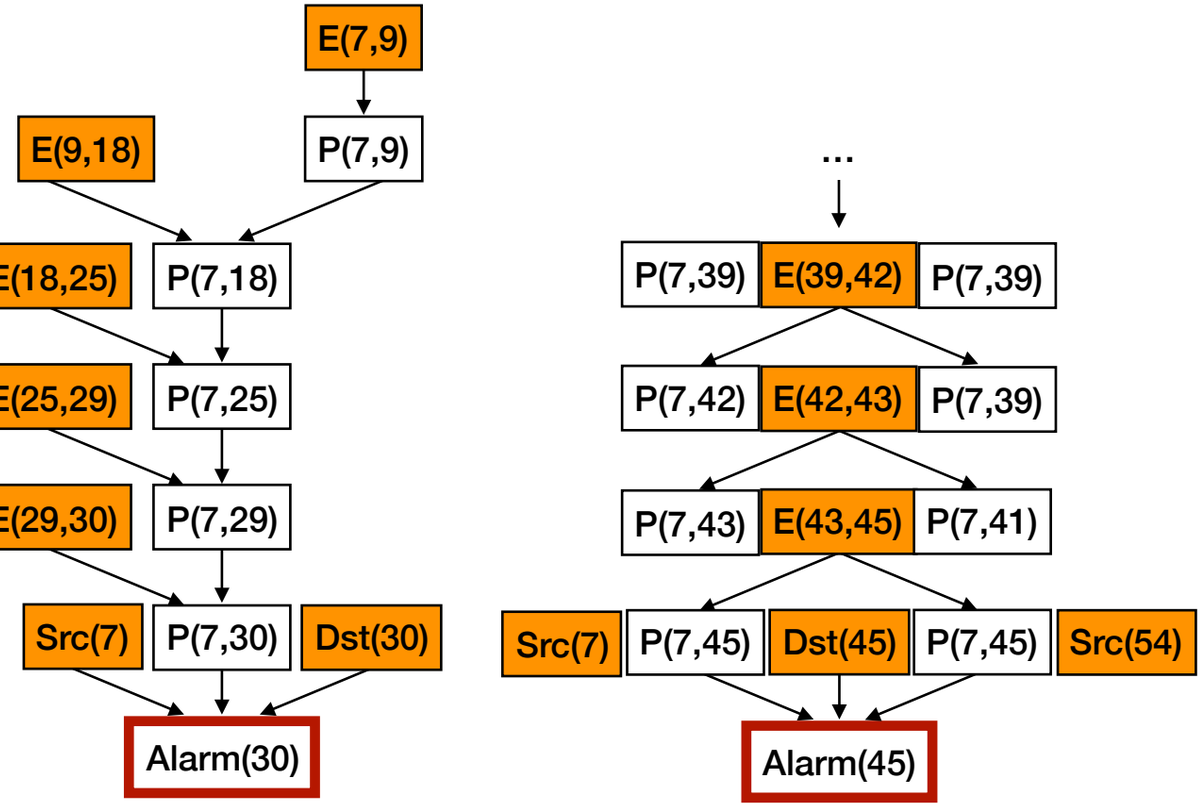


Differential Reasoning

Analysis Results of the **Old** Version

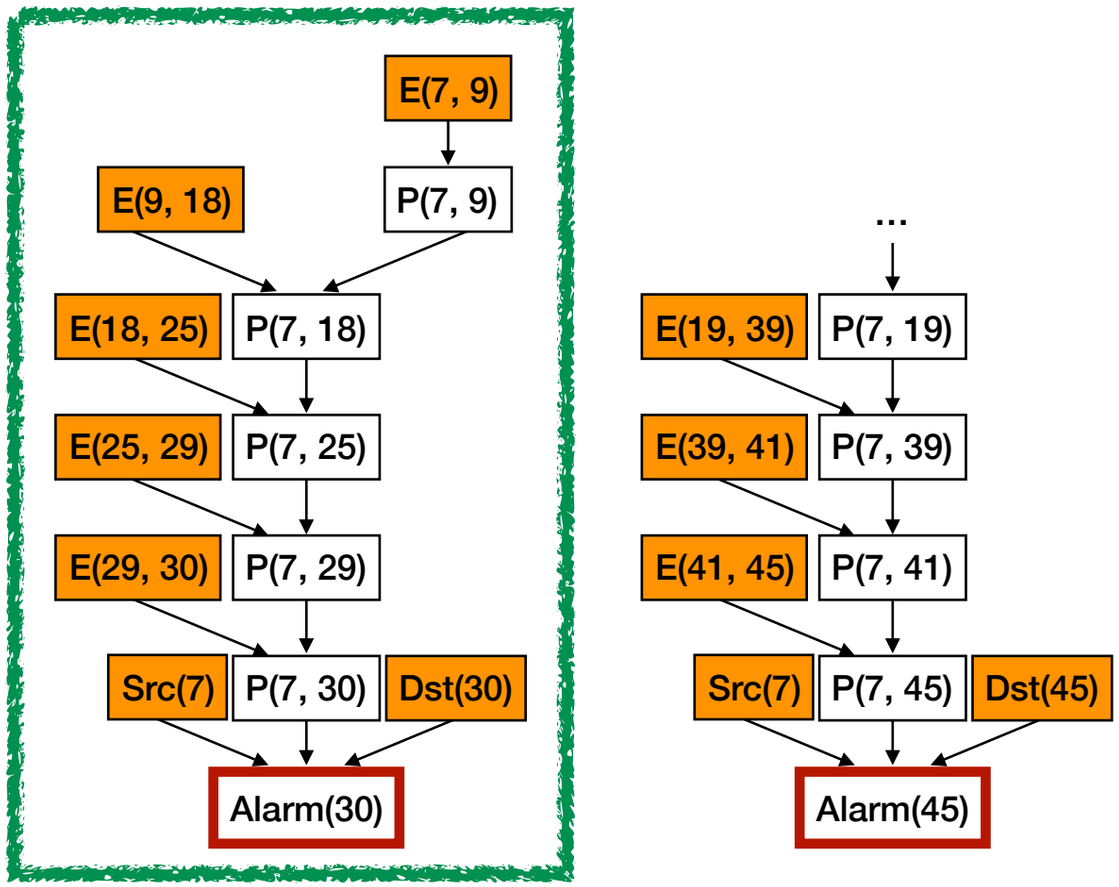


Analysis Results of the **New** Version

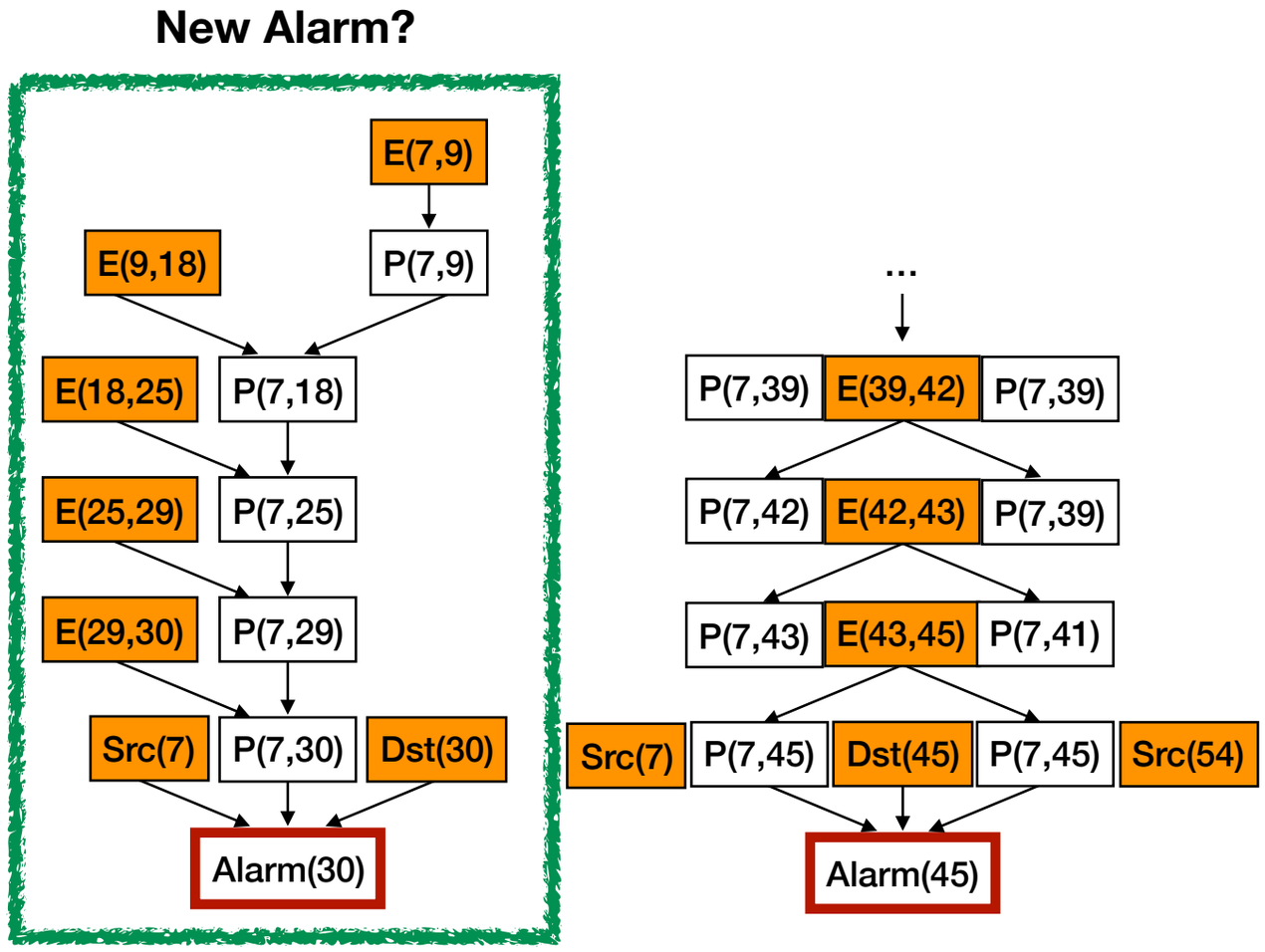


Differential Reasoning

Analysis Results of the Old Version

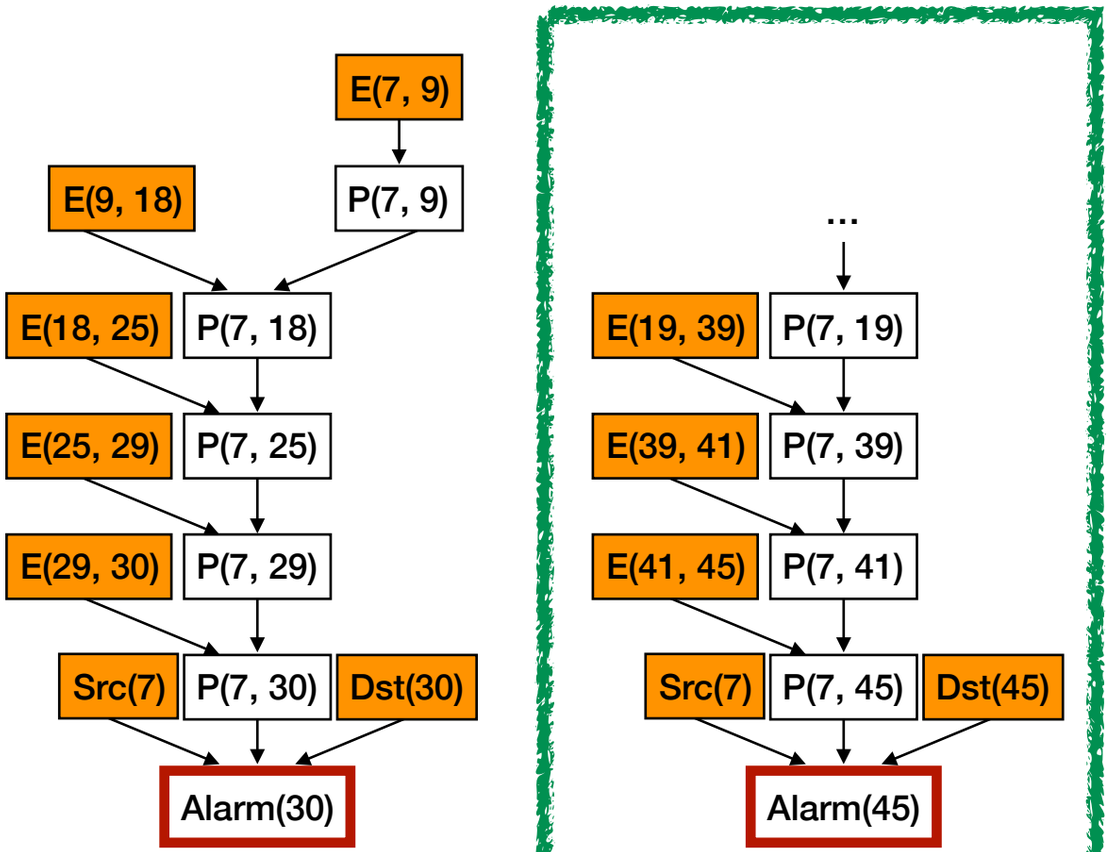


Analysis Results of the New Version

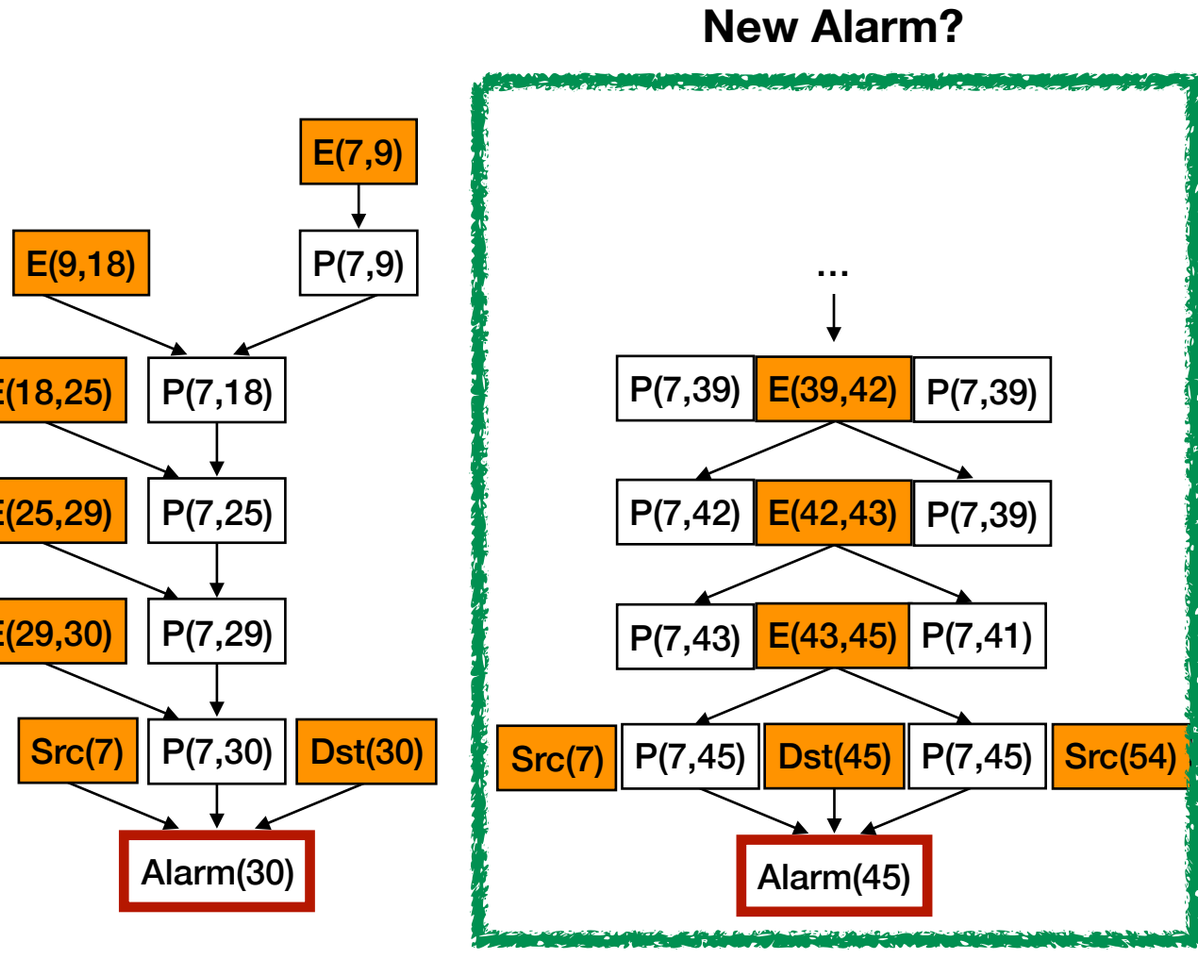


Differential Reasoning

Analysis Results of the Old Version



Analysis Results of the New Version



Challenges

Challenges

- **Semantic** alarm masking rather than syntactic

Challenges

- **Semantic** alarm masking rather than syntactic

- **Relation** between abstract states of two program versions

Challenges

- **Semantic** alarm masking rather than syntactic

- **Relation** between abstract states of two program versions

- **Relevance** of each alarm to the program change

Challenges

- **Semantic** alarm masking rather than syntactic

- **Relation** between abstract states of two program versions

- **Relevance** of each alarm to the program change

- **Ranking** alarms based on likelihood of relevance

Challenges

- **Semantic** alarm masking rather than syntactic

  ➪ **Derivations of Alarms**

- **Relation** between abstract states of two program versions

- **Relevance** of each alarm to the program change

- **Ranking** alarms based on likelihood of relevance

Challenges

- **Semantic** alarm masking rather than syntactic

  ➪ **Derivations of Alarms**

- **Relation** between abstract states of two program versions

  ➪ **Syntactic Matching Function**

- **Relevance** of each alarm to the program change

- **Ranking** alarms based on likelihood of relevance

Challenges

- **Semantic** alarm masking rather than syntactic

  ➪ **Derivations of Alarms**

- **Relation** between abstract states of two program versions

  ➪ **Syntactic Matching Function**

- **Relevance** of each alarm to the program change

  ➪ **Differential Derivation Graph**

- **Ranking** alarms based on likelihood of relevance

Challenges

- **Semantic** alarm masking rather than syntactic

  ➪ **Derivations of Alarms**

- **Relation** between abstract states of two program versions

  ➪ **Syntactic Matching Function**

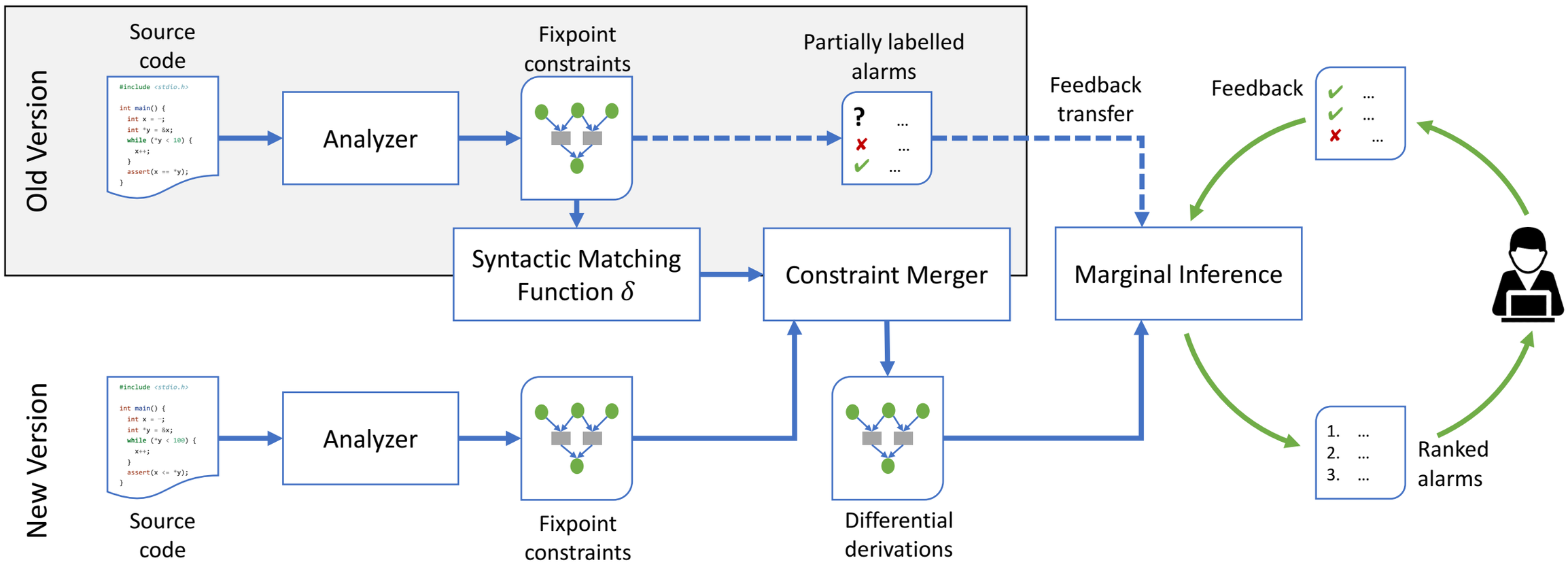
- **Relevance** of each alarm to the program change

  ➪ **Differential Derivation Graph**

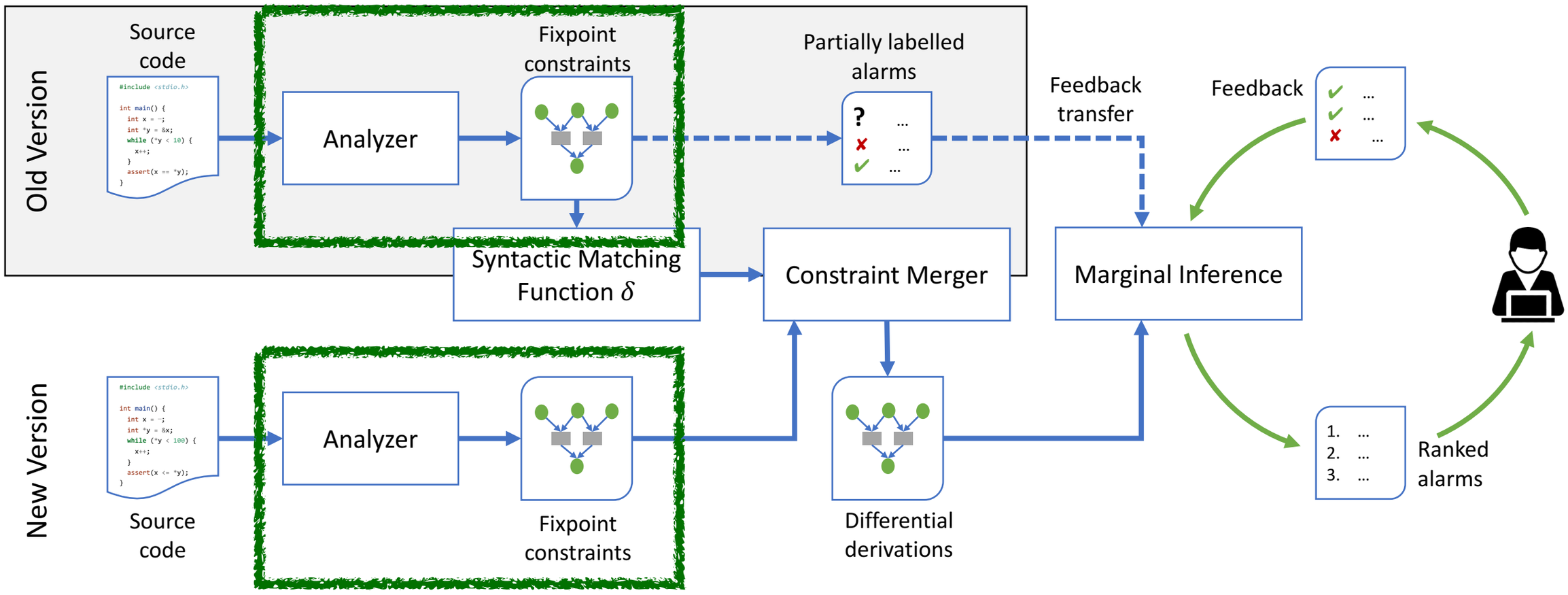
- **Ranking** alarms based on likelihood of relevance

  ➪ **Bayesian Inference**

System Architecture



System Architecture

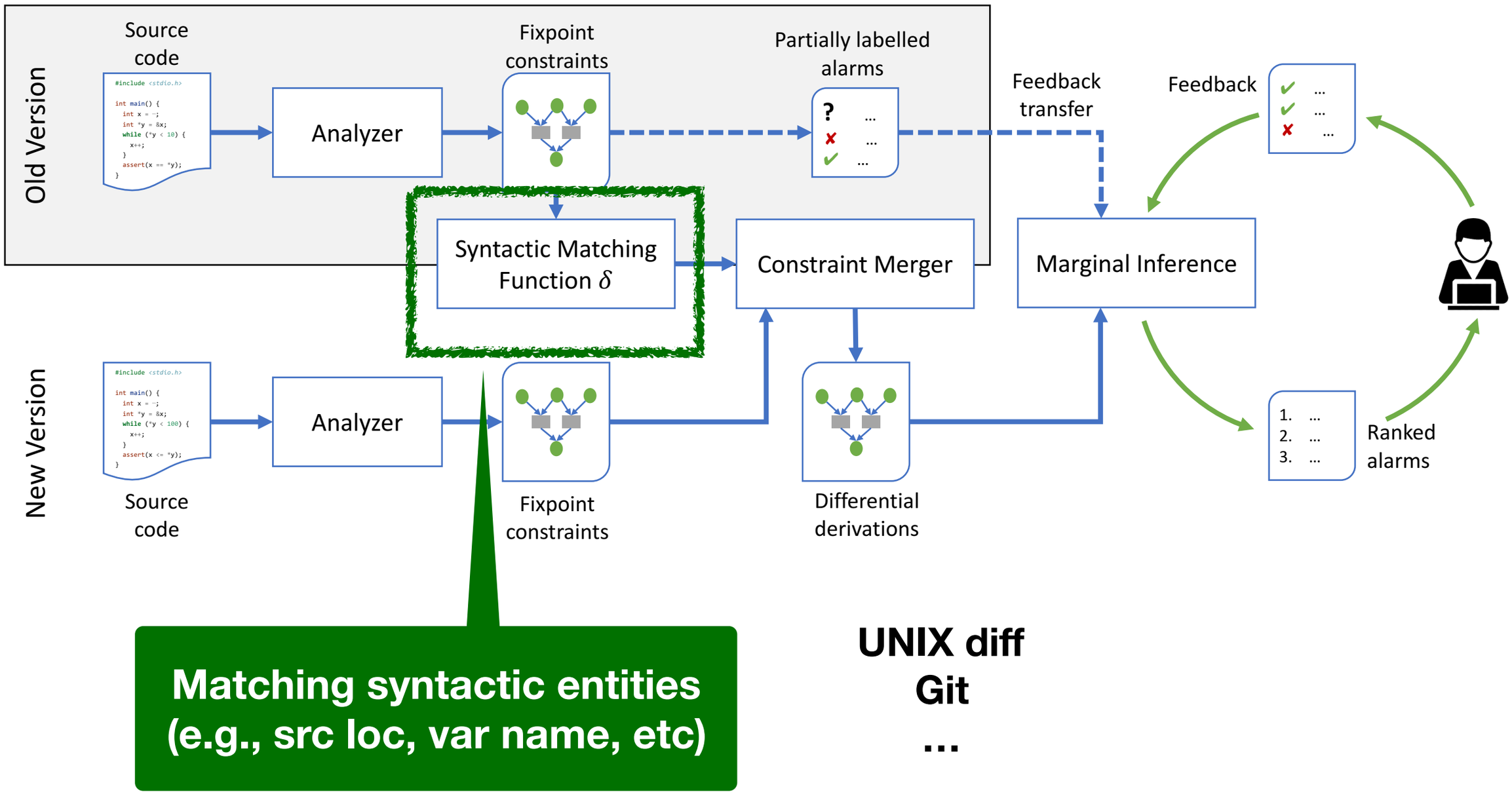


Program analyses based on deductive reasoning

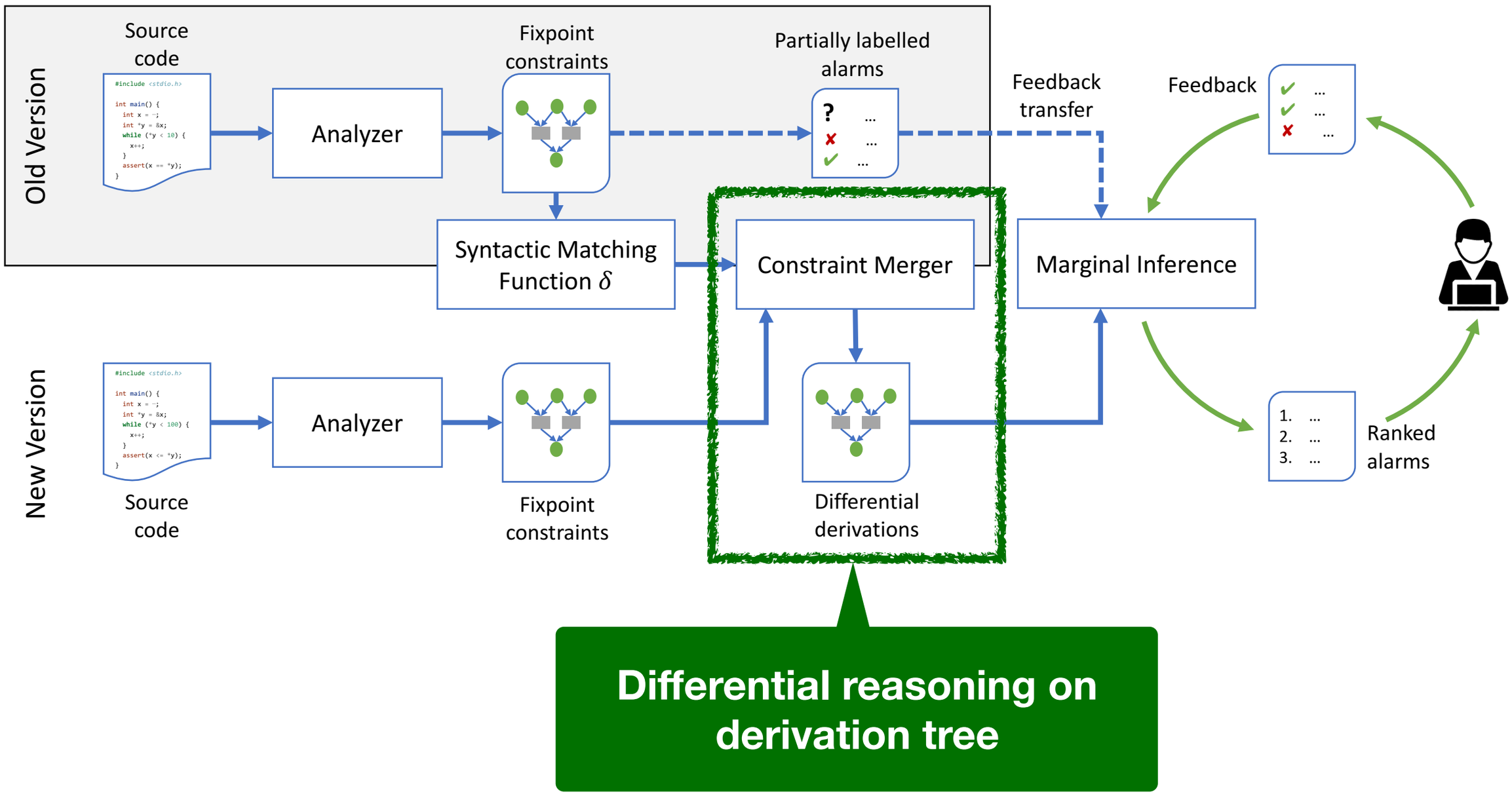
Sparrow The Early Bird

PETABLOX

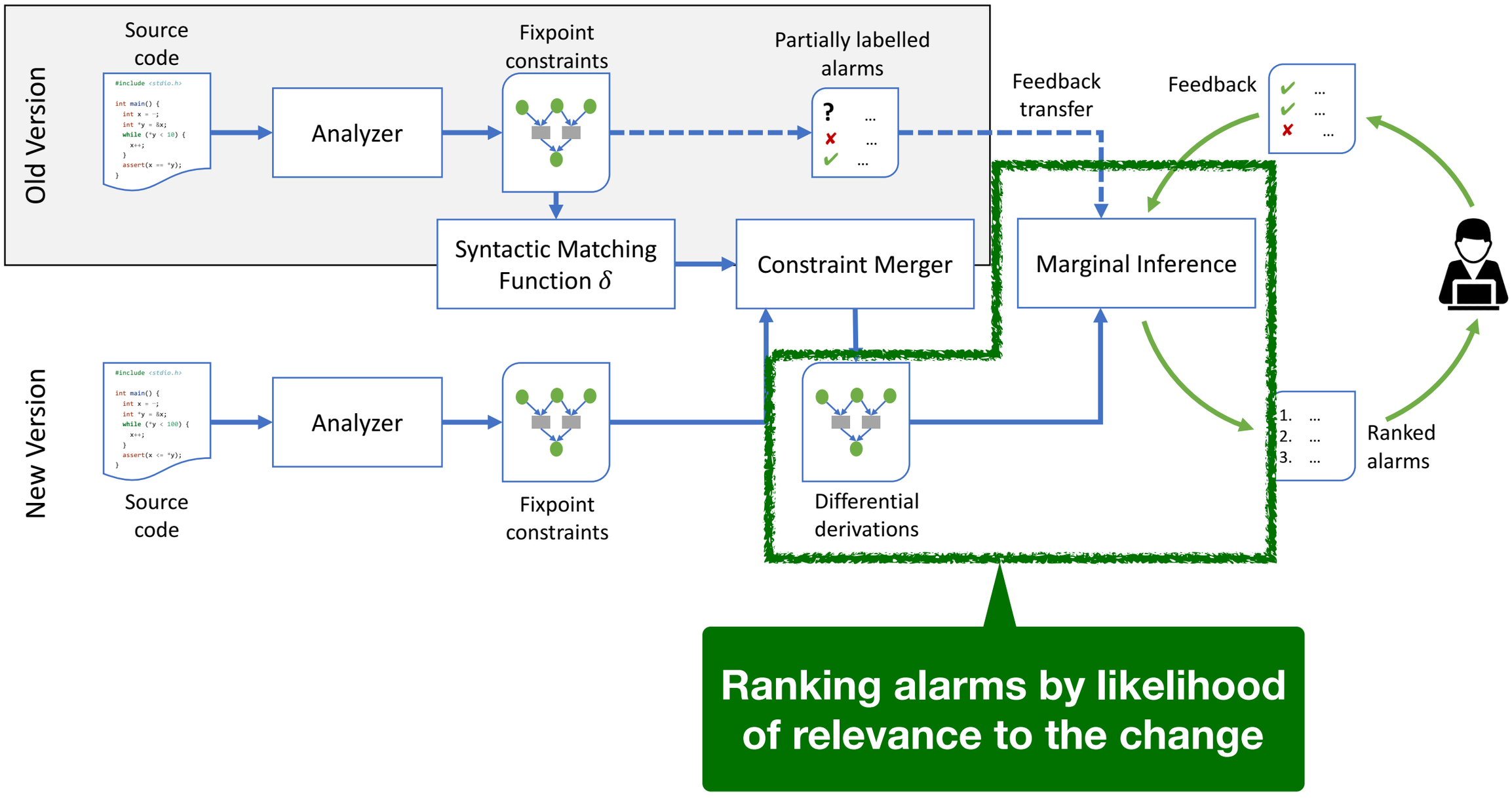
System Architecture



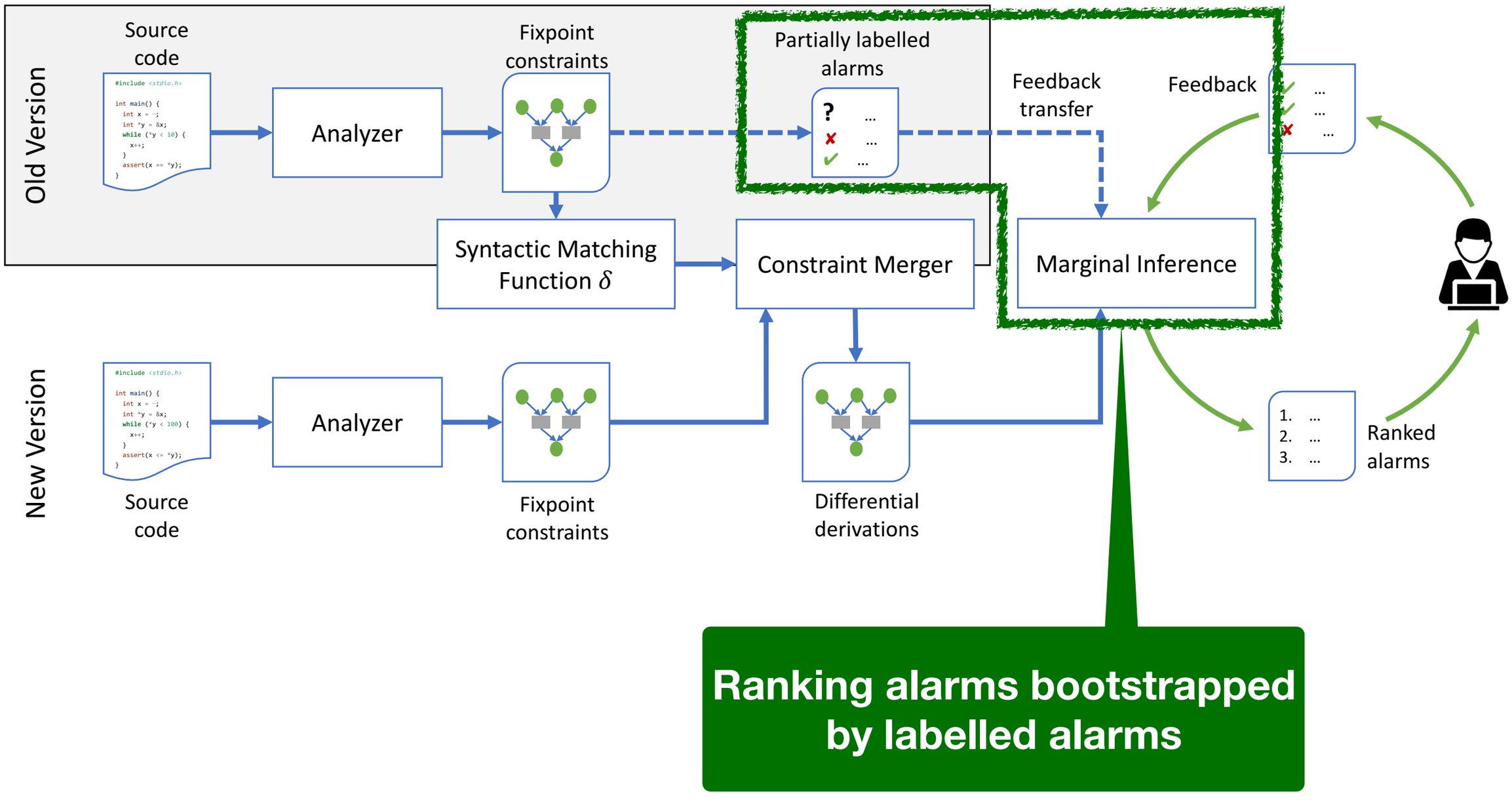
System Architecture



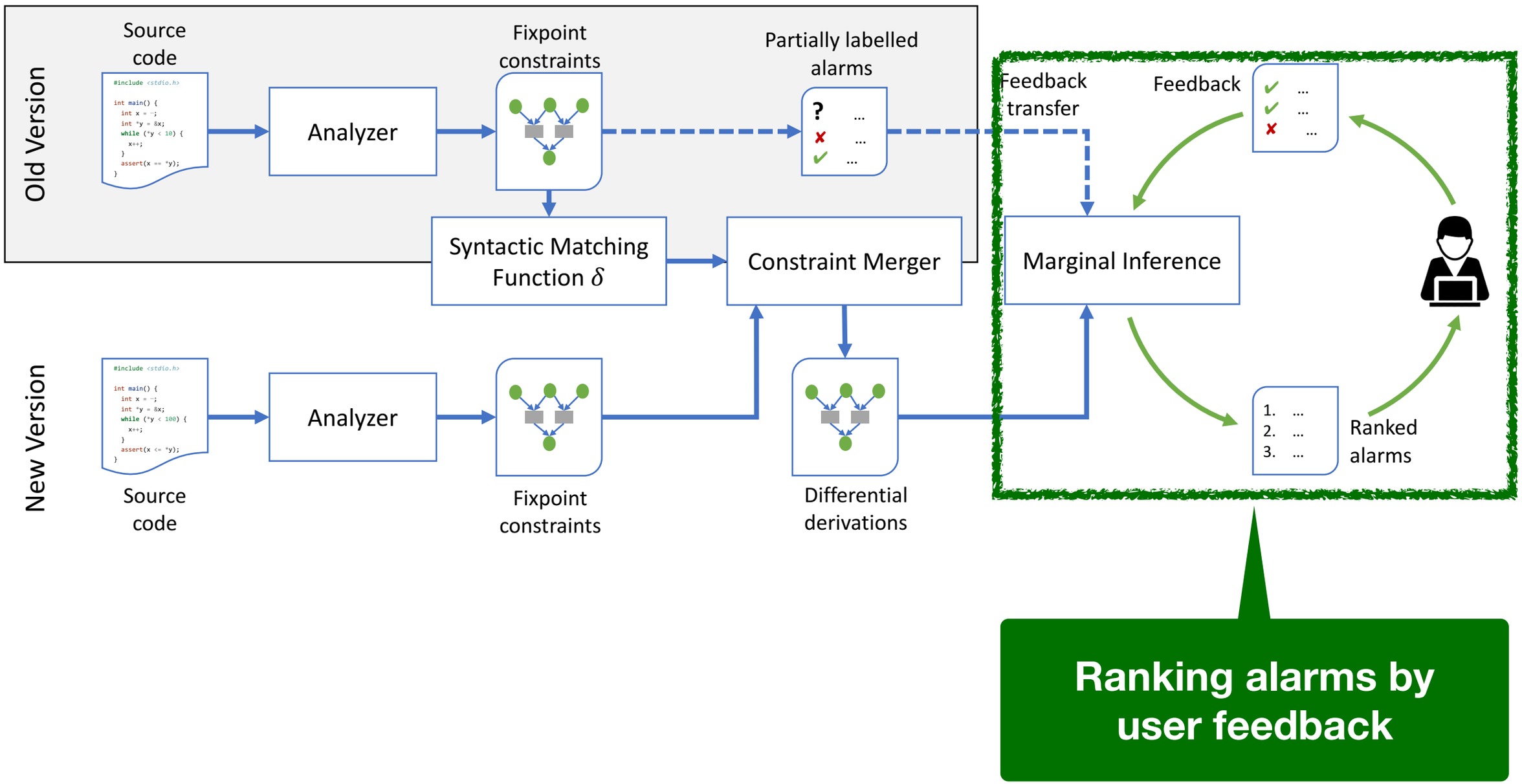
System Architecture



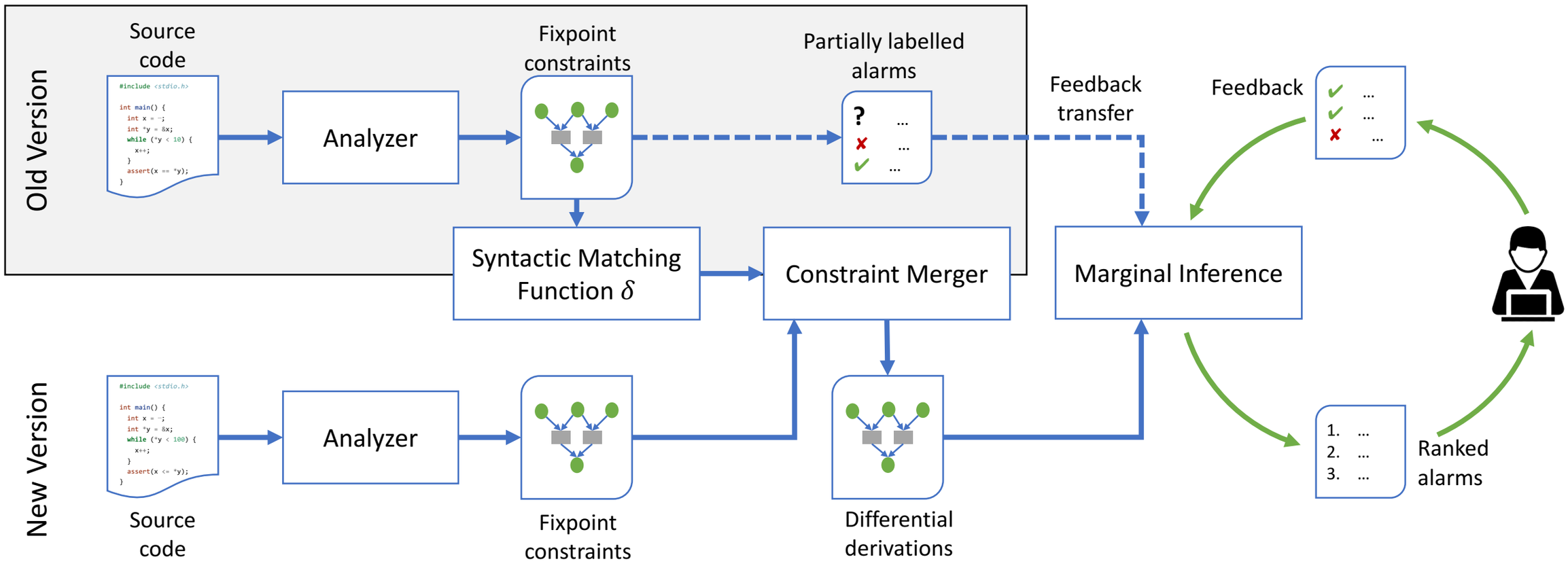
System Architecture



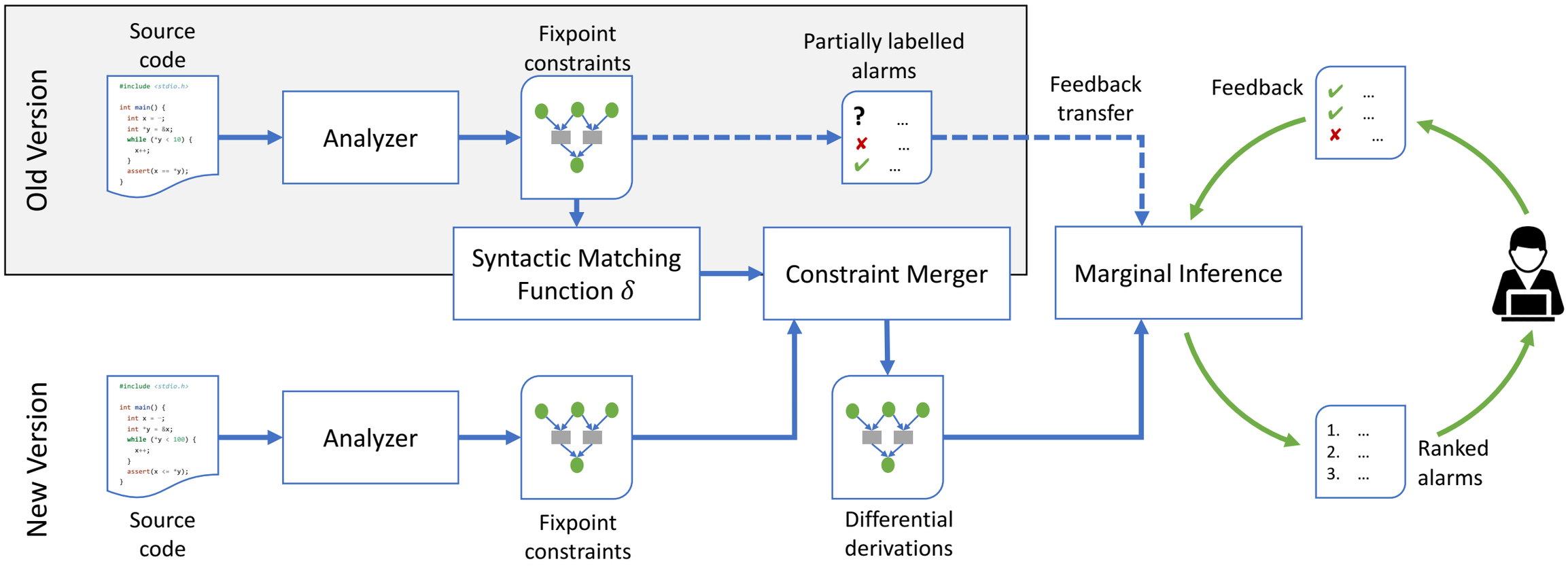
System Architecture



Impact


Old Version
Source code
#include <stdio.h>
int main() {
int x = -;
int *y = &x;
while (*y < 10) {
x++;
}
assert(x == *y);
}
Analyzer
Fixpoint constraints
Partially labelled alarms
Feedback transfer
Feedback
Syntactic Matching Function δ
Constraint Merger
Marginal Inference
New Version
Source code
#include <stdio.h>
int main() {
int x = -;
int *y = &x;
while (*y < 100) {
x++;
}
assert(x <= *y);
}
Analyzer
Fixpoint constraints
Differential derivations
1. …
2. …
3. …
Ranked alarms

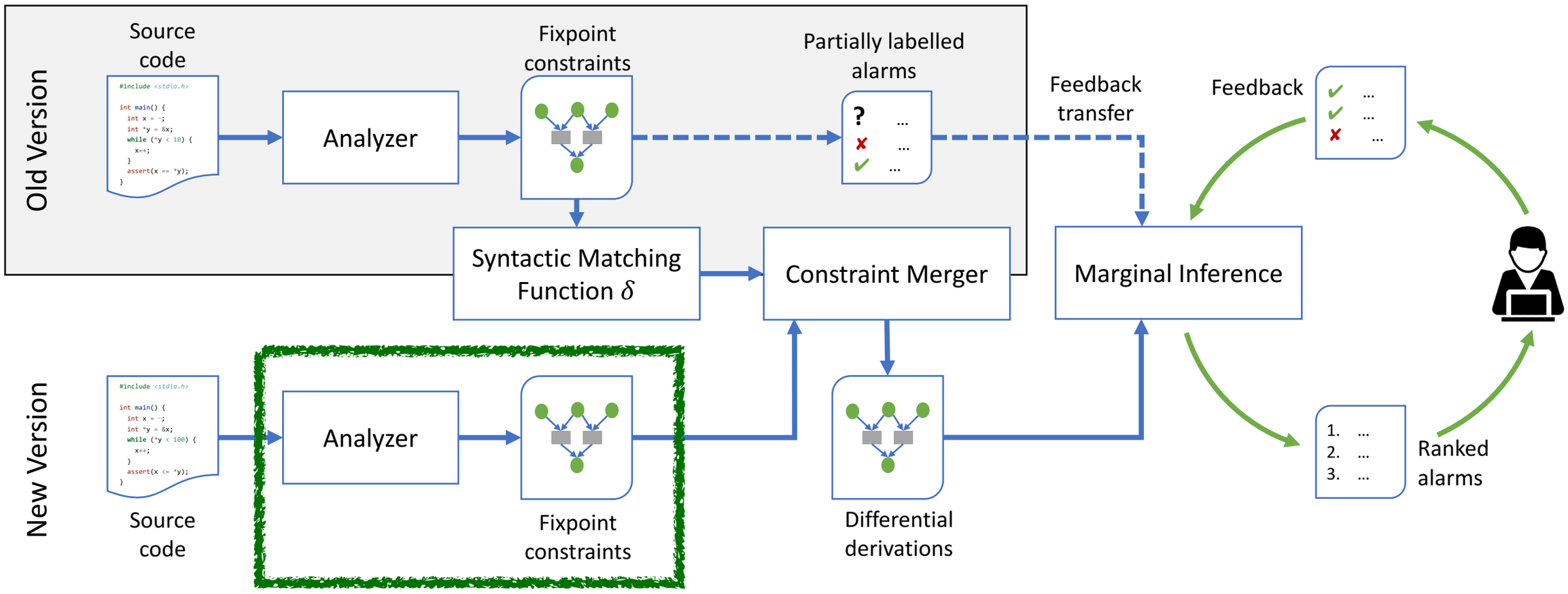
Impact



Sparrow
The Early Bird

Agv. alarms or max iters

Impact



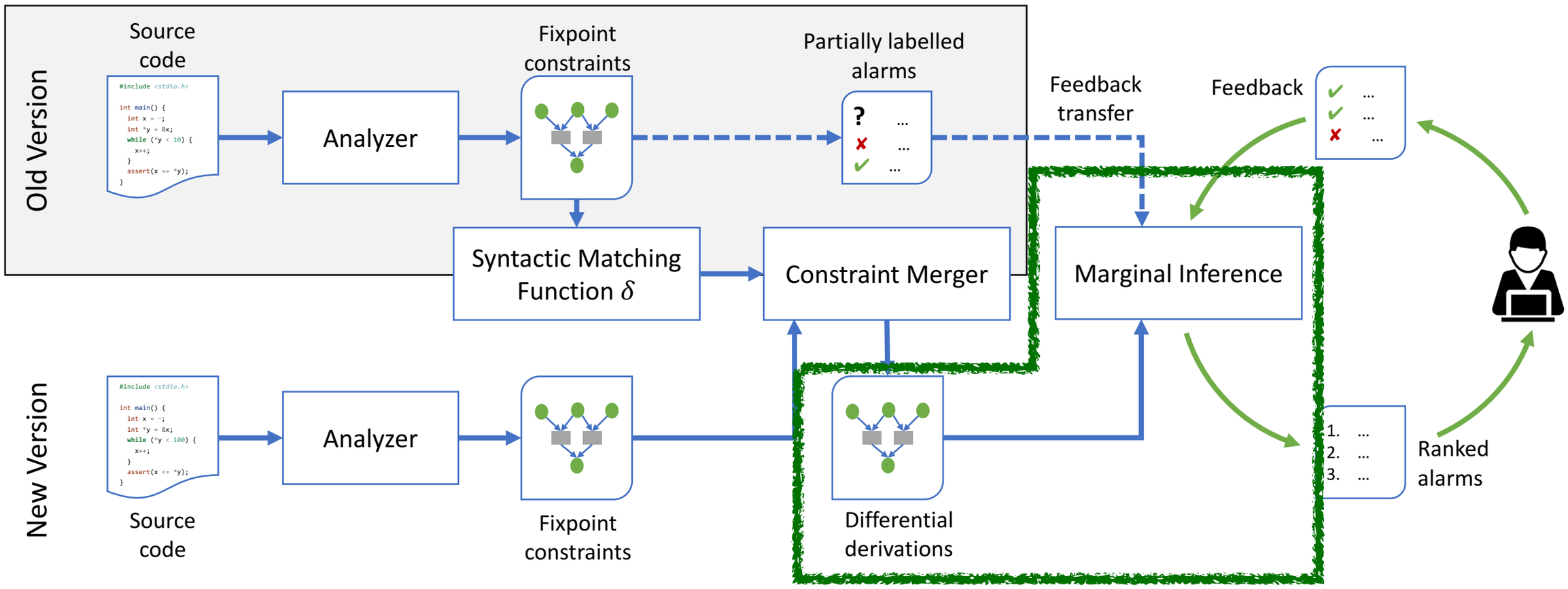
Sparrow The Early Bird

Batch mode

Agv. alarms or max iters

563

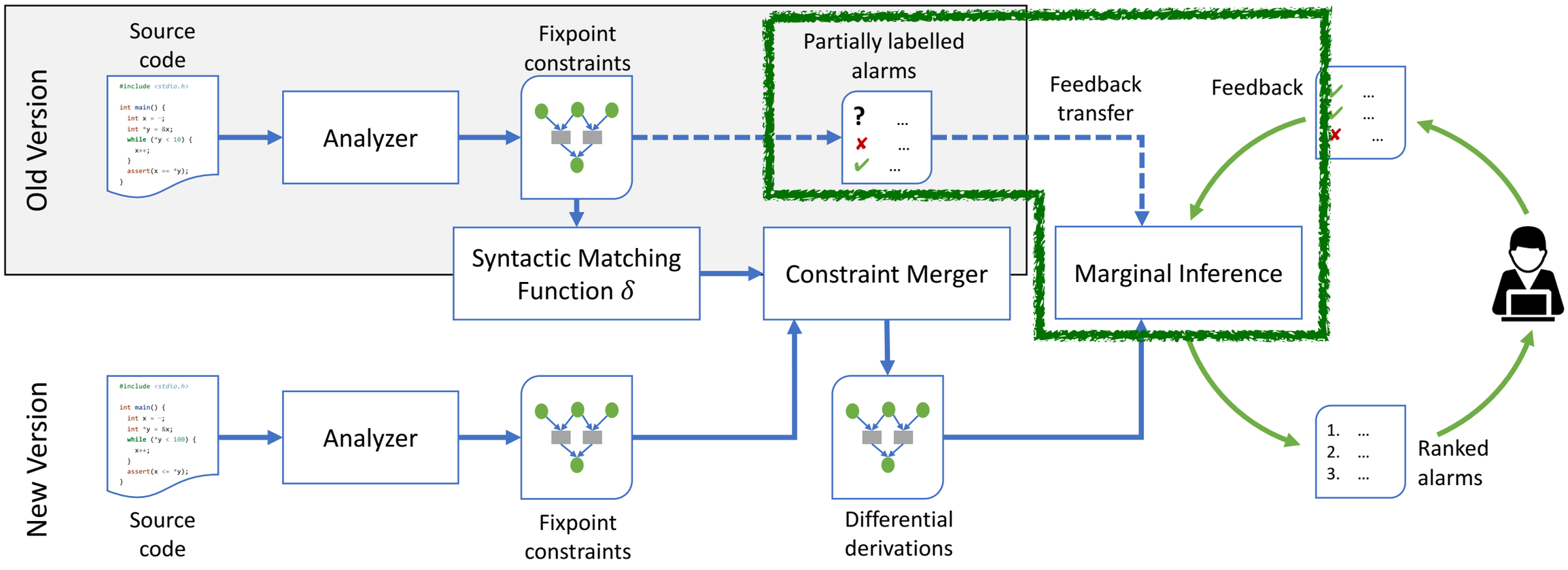
Impact



Sparrow The Early Bird

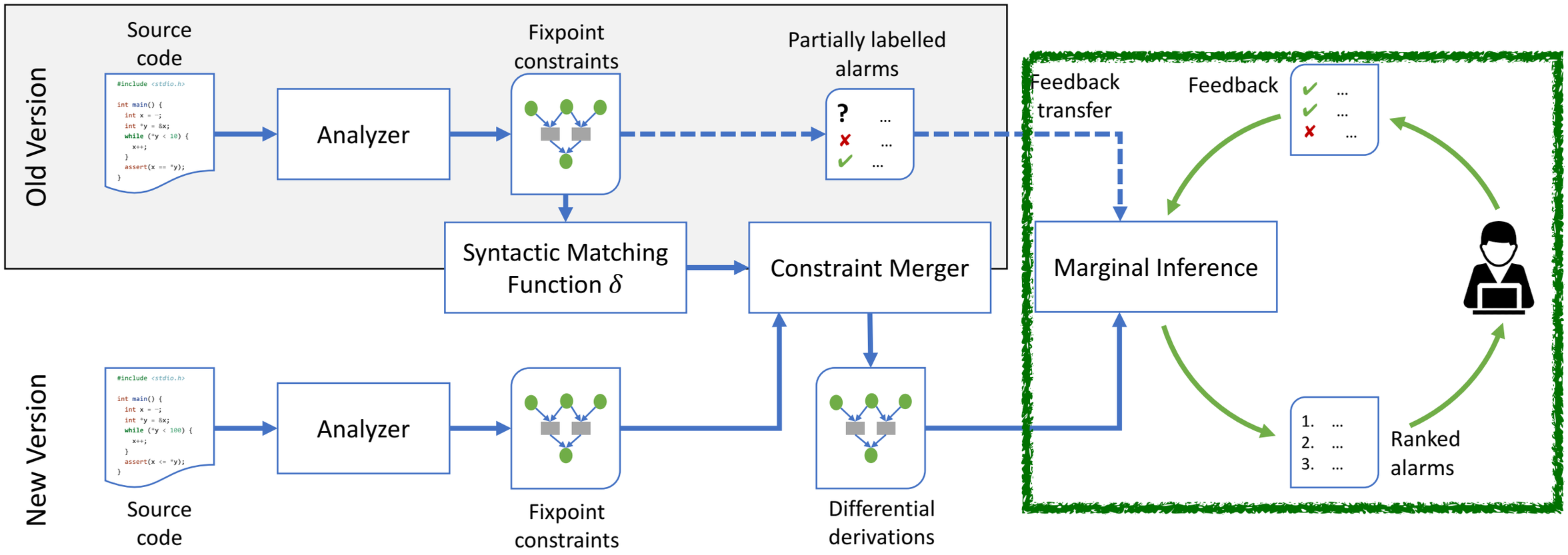
	Batch mode	Ranking by relevance
Agv. alarms or max iters	563	94

Impact



	Batch mode	Ranking by relevance	Ranking by old labels
Agv. alarms or max iters	563	94	78

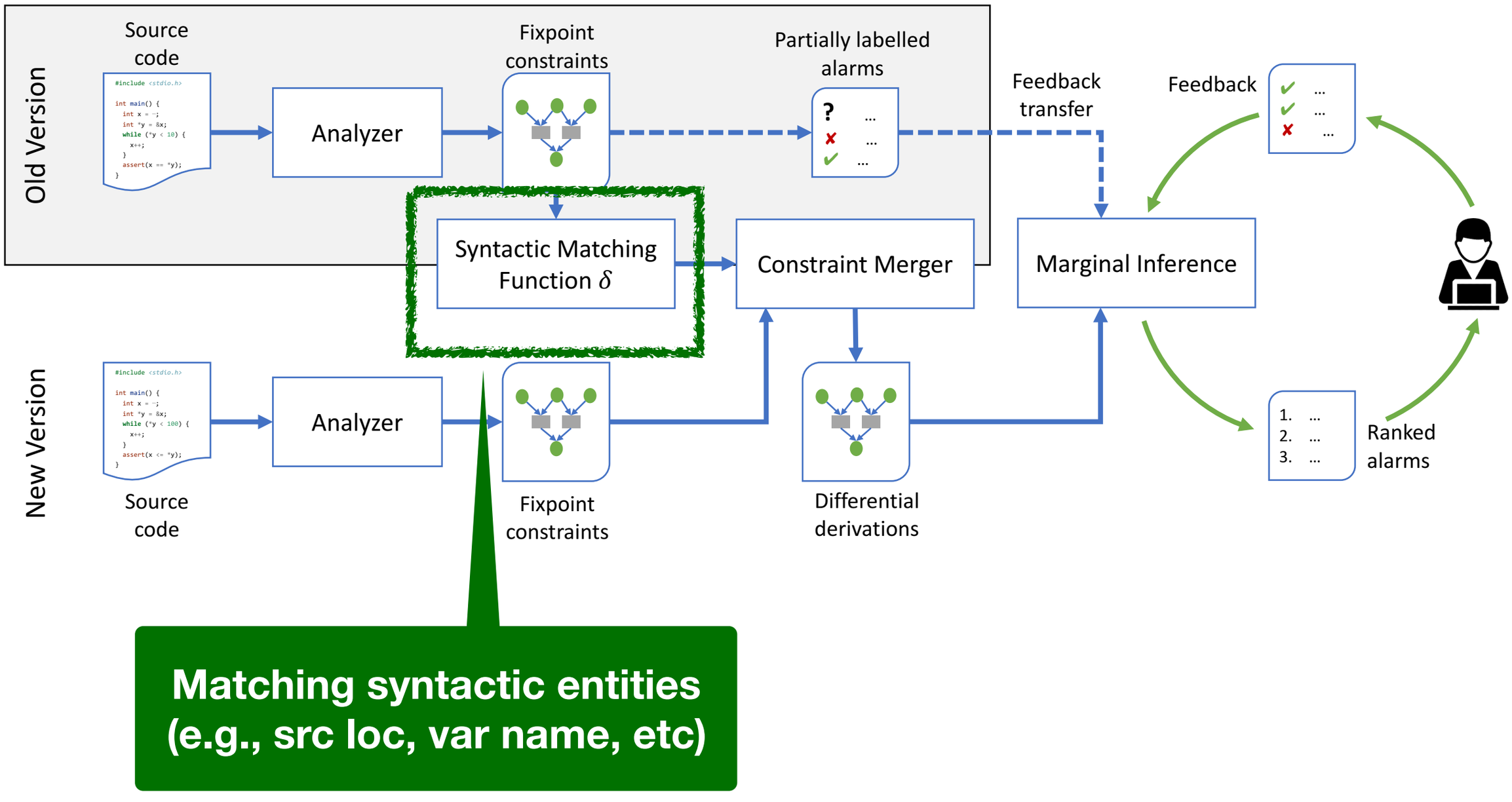
Impact



Sparrow The Early Bird

	Batch mode	Ranking by relevance	Ranking by old labels	Ranking by interaction
Agv. alarms or max iters	563	94	78	30

System Architecture



Syntactic Matching

- Many semantic components depend on syntactic entities
  - e.g., program point, allocation-site, call-site
- Syntactic matching function are parameterized
  - e.g., Unix diff (line), git (file), etc

Syntactic Matching

- Many semantic components depend on syntactic entities
  - e.g., program point, allocation-site, call-site
- Syntactic matching function are parameterized
  - e.g., Unix diff (line), git (file), etc

```
1: int main() {
2:  x = read();
3:  y = x * 4;
4:  p = malloc(y);
5: }
```

```
  1: int main() {
  2:  x = read();
+ 3:  tmp = 0;
  4:  y = x * 4;
  5:  p = malloc(y);
  6: }
```

Syntactic Matching

- Many semantic components depend on syntactic entities
  - e.g., program point, allocation-site, call-site
- Syntactic matching function are parameterized
  - e.g., Unix diff (line), git (file), etc

```
1: int main() {
2:   x = read();
3:   y = x * 4;
4:   p = malloc(y);
5: }
```

```
1: int main() {
2:   x = read();
+ 3:   tmp = 0;
4:   y = x * 4;
5:   p = malloc(y);
6: }
```

Syntactic Matching

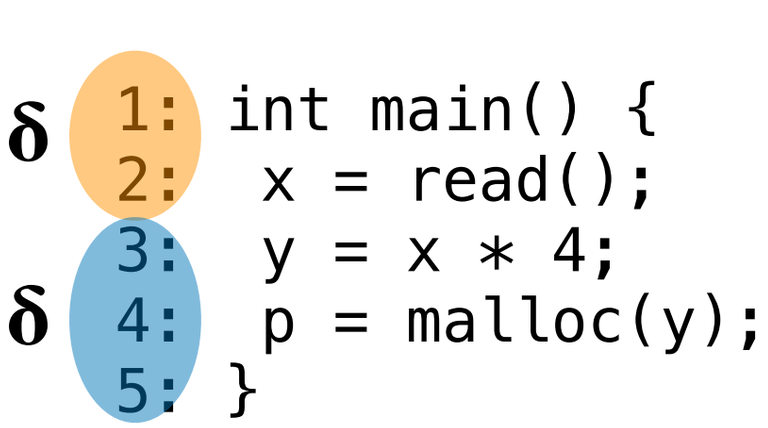
- Many semantic components depend on syntactic entities
  - e.g., program point, allocation-site, call-site
- Syntactic matching function are parameterized
  - e.g., Unix diff (line), git (file), etc

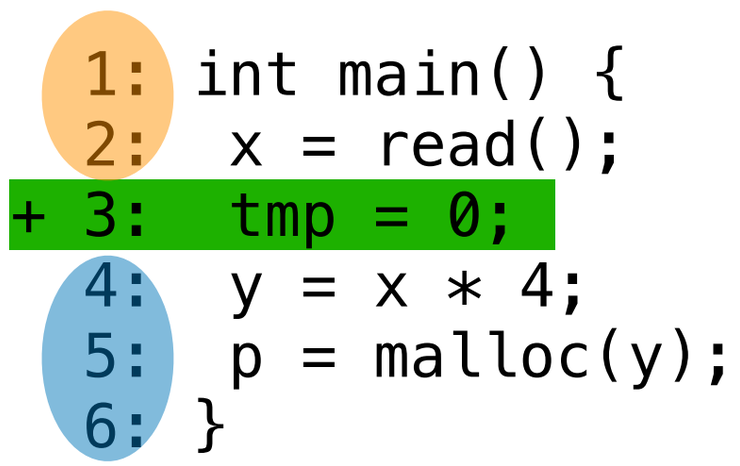
```
1: int main() {
2:   x = read();
3:   y = x * 4;
4:   p = malloc(y);
5: }
```

```
1: int main() {
2:   x = read();
+ 3:   tmp = 0;
4:   y = x * 4;
5:   p = malloc(y);
6: }
```

Syntactic Matching

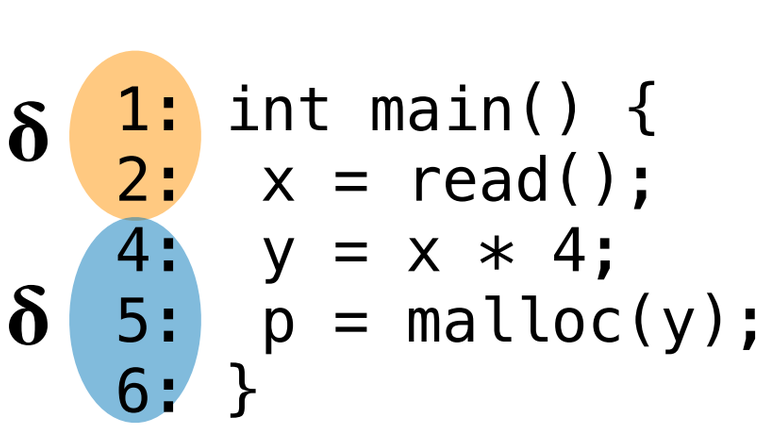
- Many semantic components depend on syntactic entities
  - e.g., program point, allocation-site, call-site
- Syntactic matching function are parameterized
  - e.g., Unix diff (line), git (file), etc

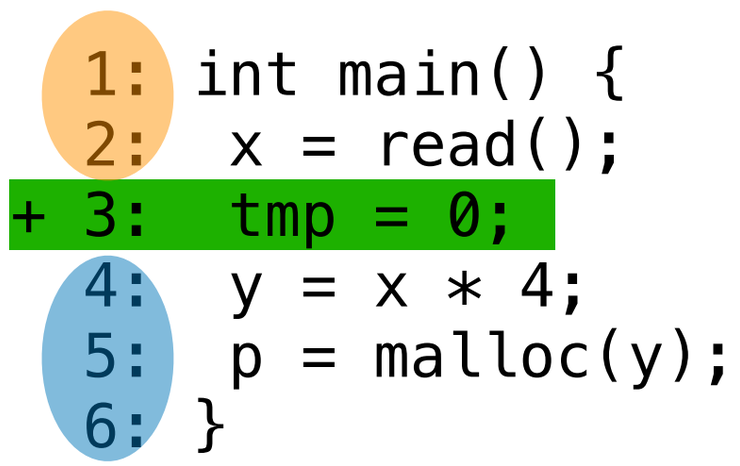




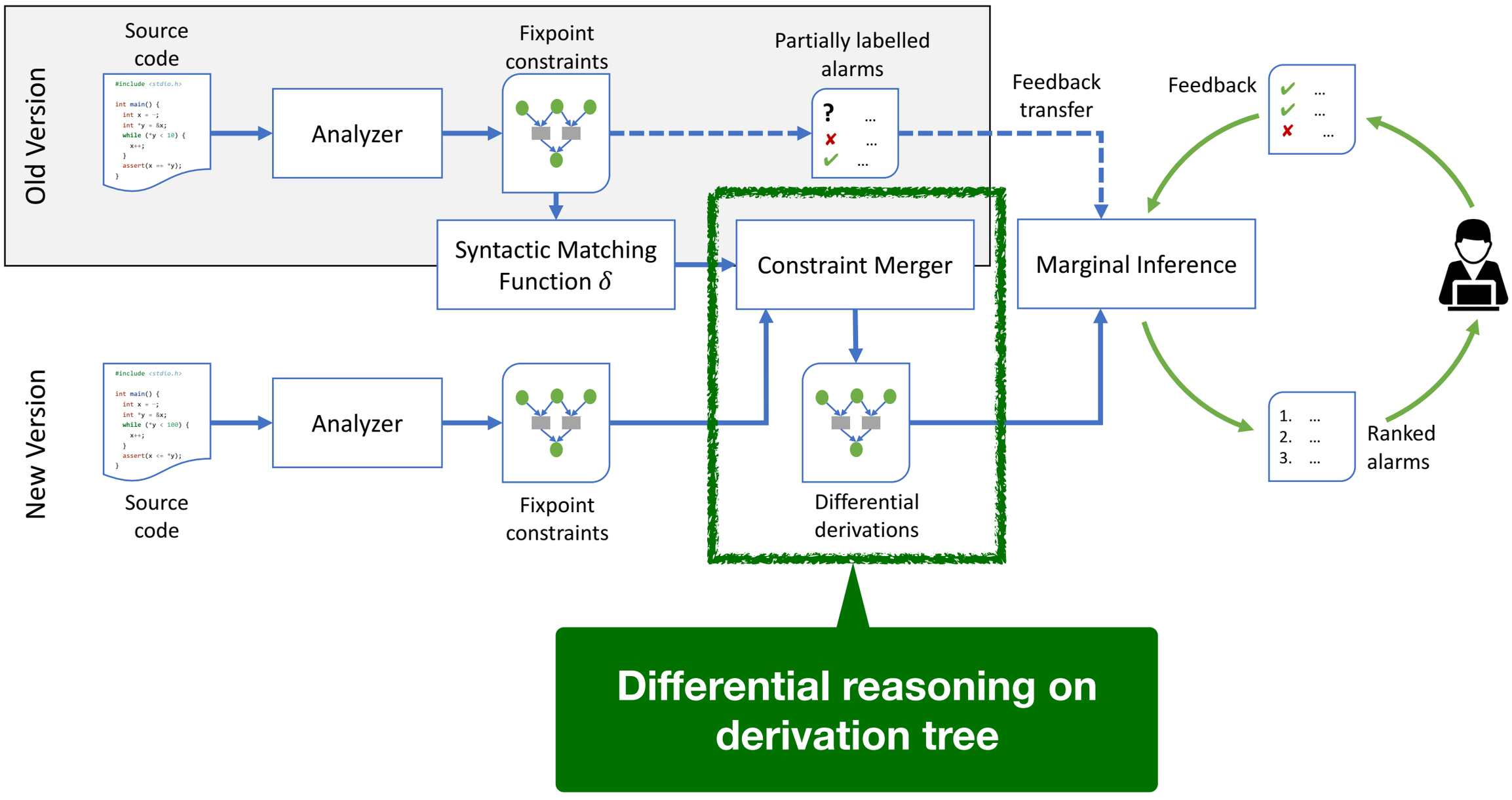
Syntactic Matching

- Many semantic components depend on syntactic entities
  - e.g., program point, allocation-site, call-site
- Syntactic matching function are parameterized
  - e.g., Unix diff (line), git (file), etc





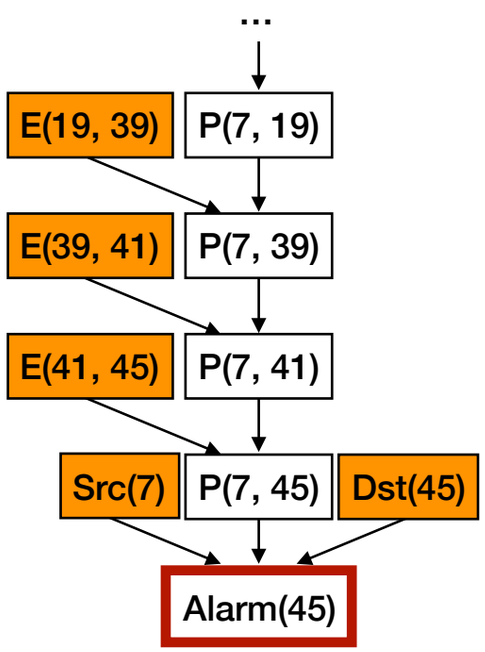
System Architecture



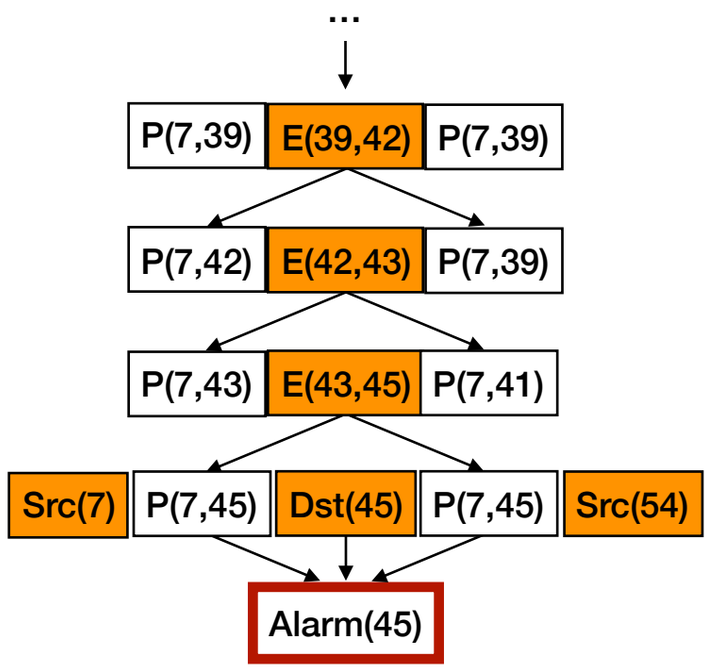
Differential Derivation

Q: Does this alarm have at least one new derivation?

Old Analysis



New Analysis



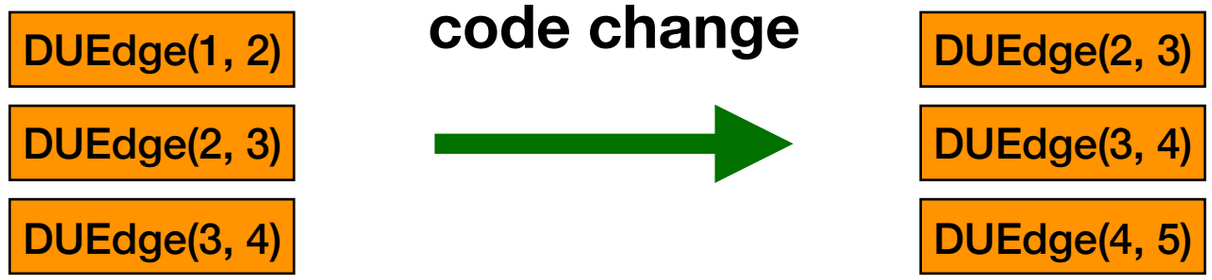
Differential Derivation

Plan: Construct differential derivation inductively

Differential Derivation

Plan: Construct differential derivation inductively

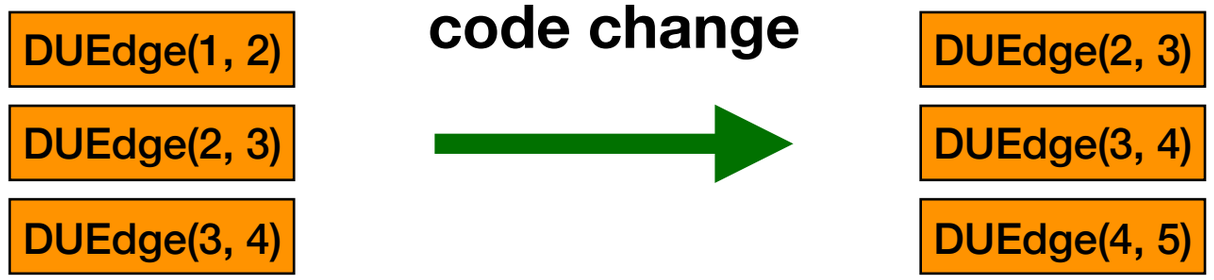
1. Base case: Input relations



Differential Derivation

Plan: Construct differential derivation inductively

1. Base case: Input relations



***α*:** all derivations are **common** to both versions

***β*:** at least one **new** derivation exists

Differential Derivation

Plan: Construct differential derivation inductively

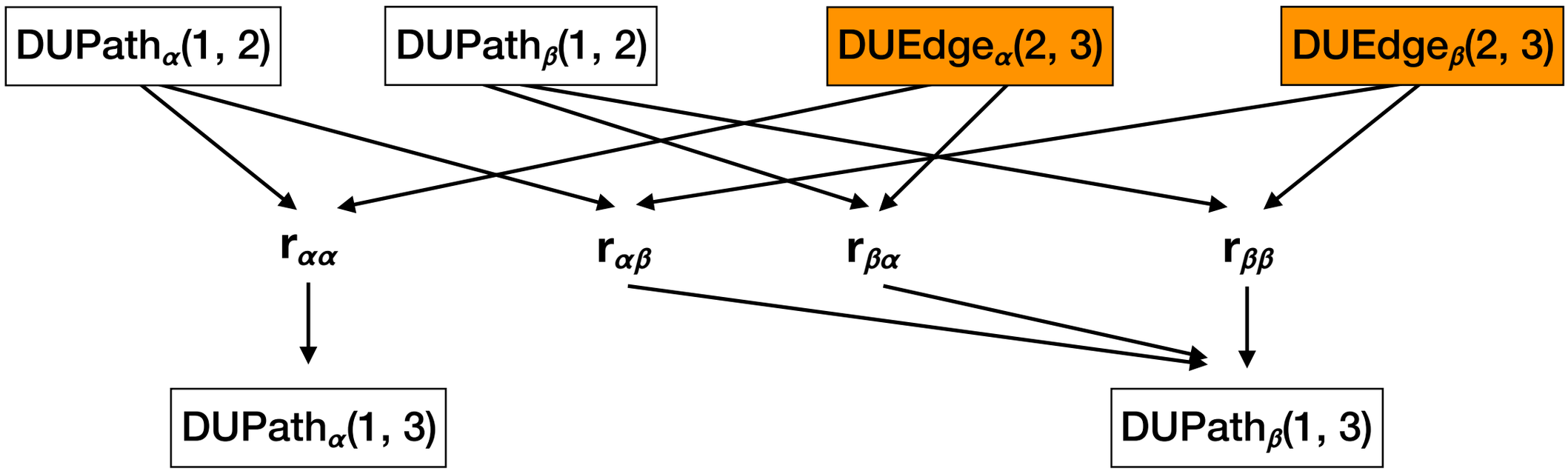
2. Inductive case: Output relations

Differential Derivation

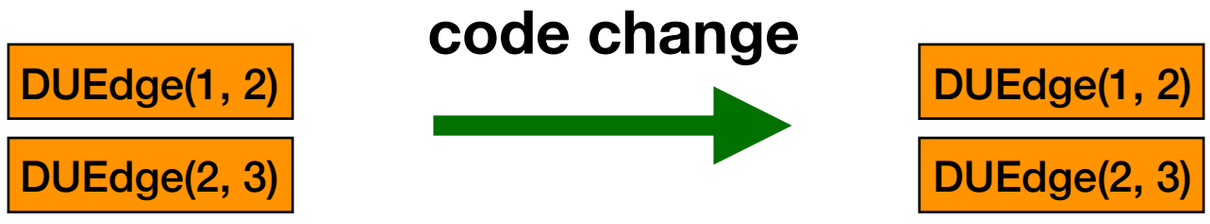
Plan: Construct differential derivation inductively

2. Inductive case: Output relations

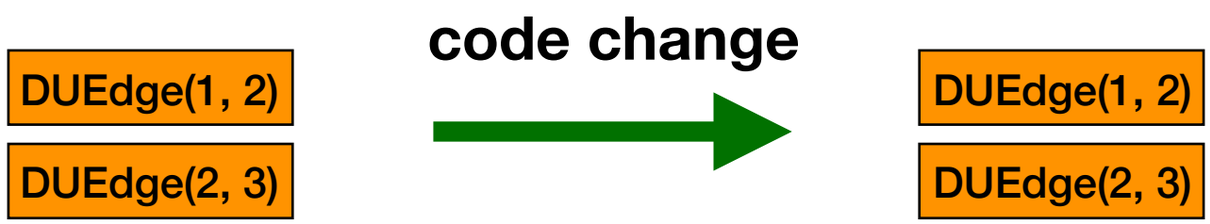
***r*: DUPath(c1, c3) :- DUPath(c1, c2), DUEdge(c2, c3)**

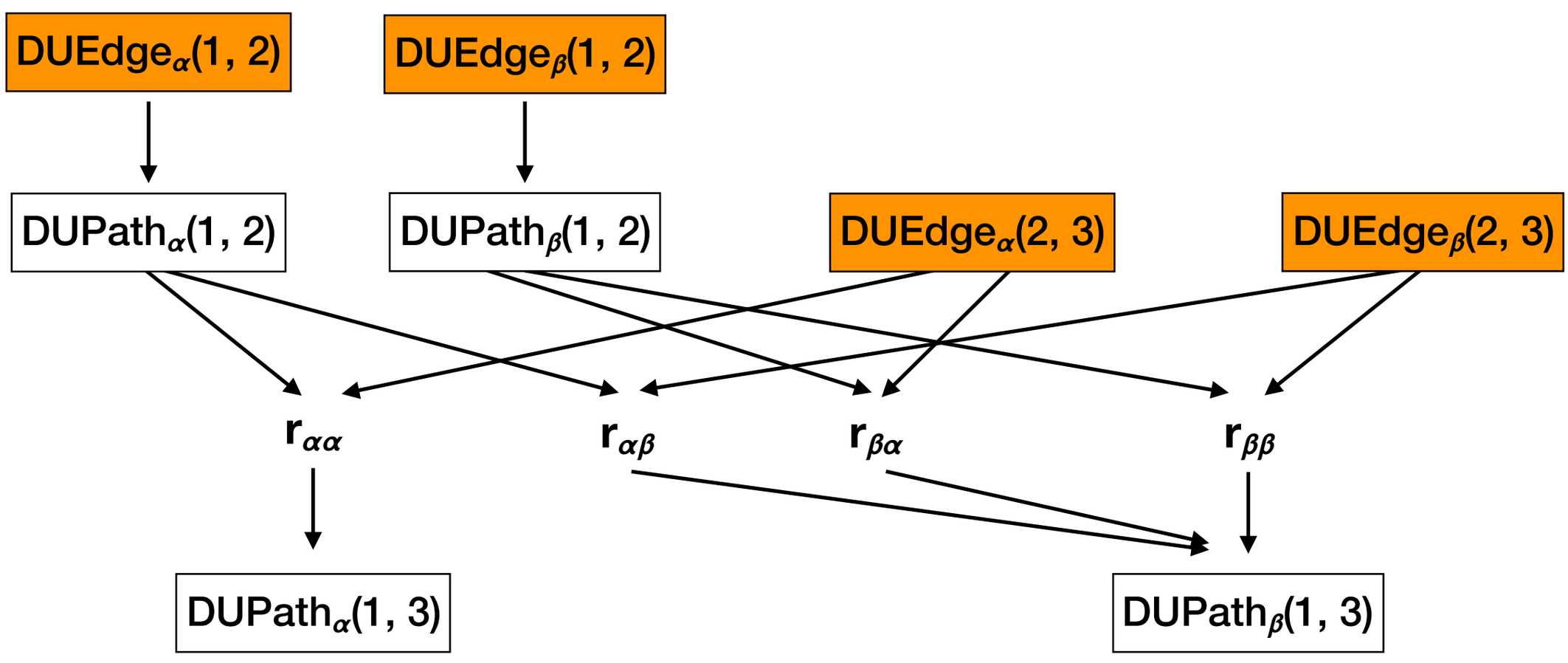


Example

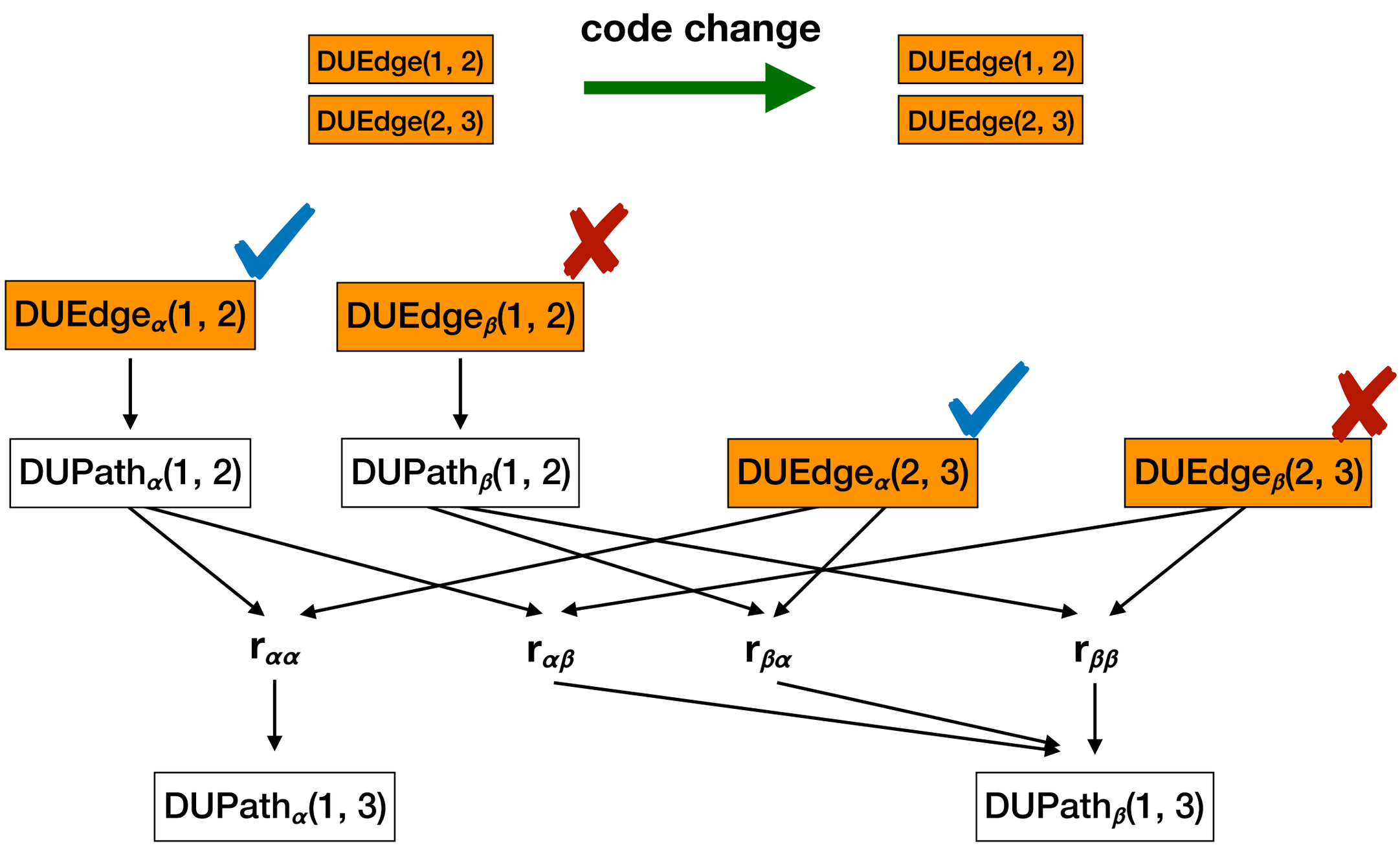


Example

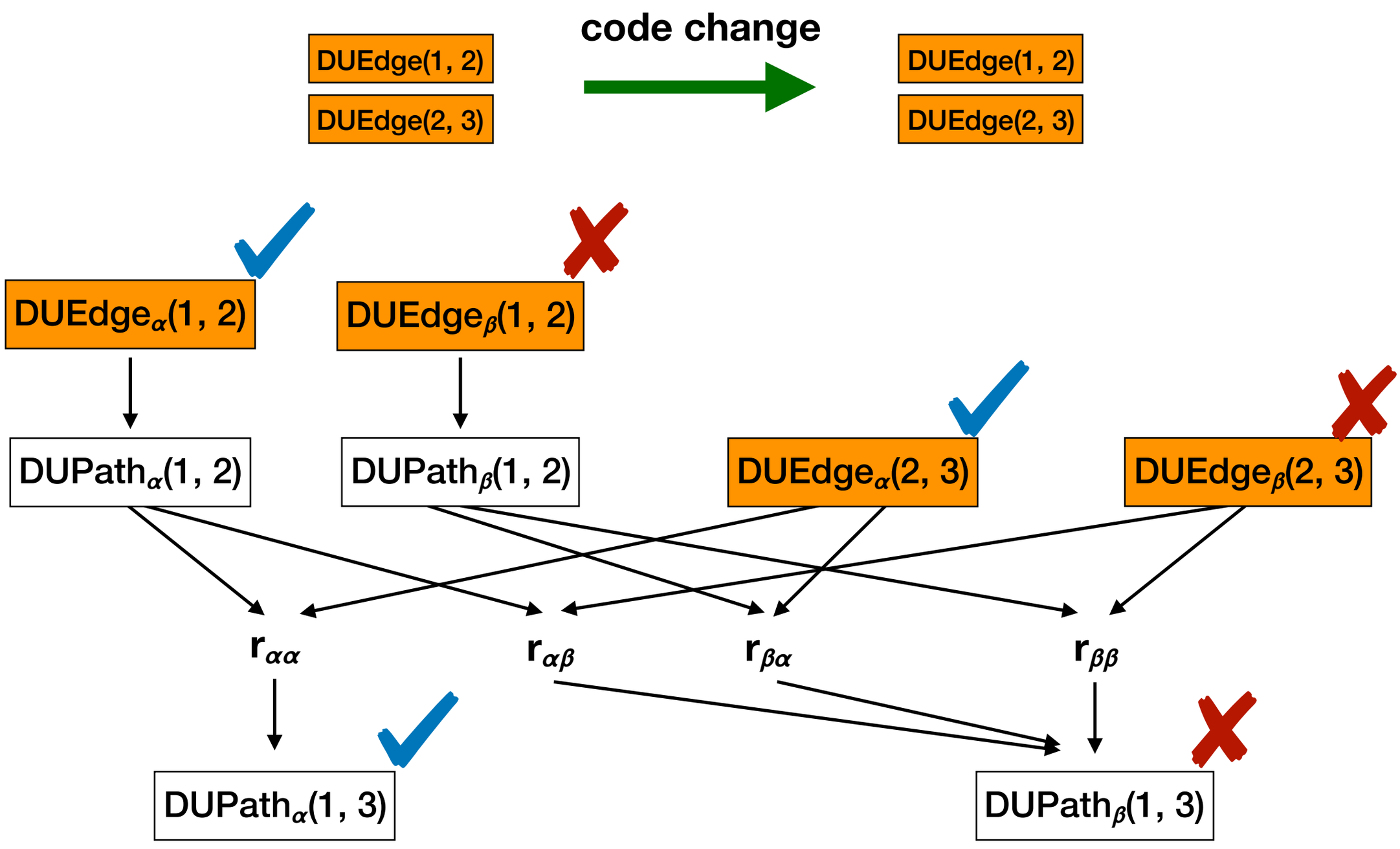

DUEdge(1, 2)
DUEdge(2, 3)
code change
DUEdge(1, 2)
DUEdge(2, 3)


DUEdge$_\alpha$(1, 2)
DUEdge$_\beta$(1, 2)
DUPath$_\alpha$(1, 2)
DUPath$_\beta$(1, 2)
DUEdge$_\alpha$(2, 3)
DUEdge$_\beta$(2, 3)
r$_{\alpha\alpha}$
r$_{\alpha\beta}$
r$_{\beta\alpha}$
r$_{\beta\beta}$
DUPath$_\alpha$(1, 3)
DUPath$_\beta$(1, 3)

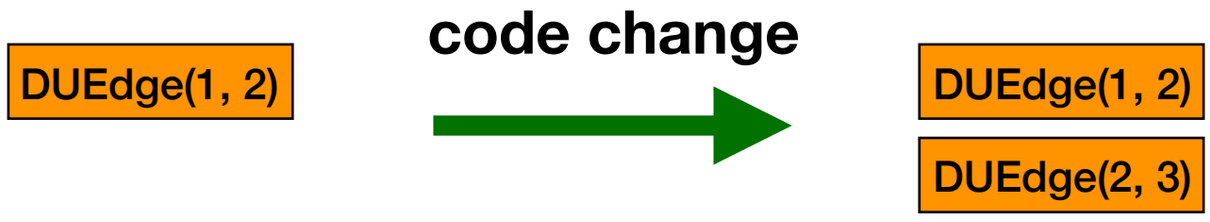
Example


DUEdge(1, 2)
DUEdge(2, 3)
code change
DUEdge(1, 2)
DUEdge(2, 3)
DUEdge$_\alpha$(1, 2)
DUEdge$_\beta$(1, 2)
DUPath$_\alpha$(1, 2)
DUPath$_\beta$(1, 2)
DUEdge$_\alpha$(2, 3)
DUEdge$_\beta$(2, 3)
r$_{\alpha\alpha}$
r$_{\alpha\beta}$
r$_{\beta\alpha}$
r$_{\beta\beta}$
DUPath$_\alpha$(1, 3)
DUPath$_\beta$(1, 3)

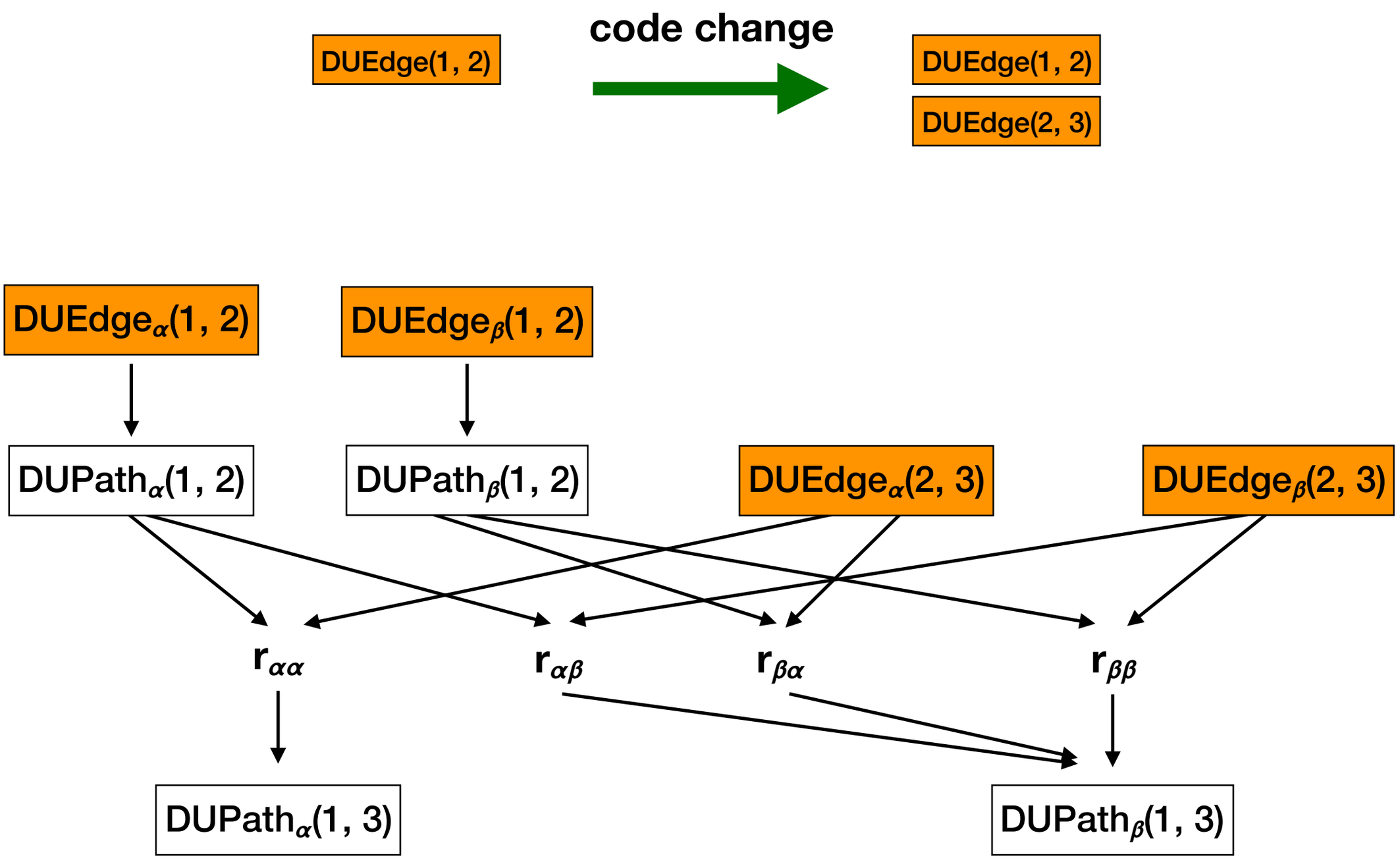
Example


DUEdge(1, 2)
DUEdge(2, 3)
code change
DUEdge(1, 2)
DUEdge(2, 3)
DUEdge$_\alpha$(1, 2)
DUEdge$_\beta$(1, 2)
DUPath$_\alpha$(1, 2)
DUPath$_\beta$(1, 2)
DUEdge$_\alpha$(2, 3)
DUEdge$_\beta$(2, 3)
r$_{\alpha\alpha}$
r$_{\alpha\beta}$
r$_{\beta\alpha}$
r$_{\beta\beta}$
DUPath$_\alpha$(1, 3)
DUPath$_\beta$(1, 3)

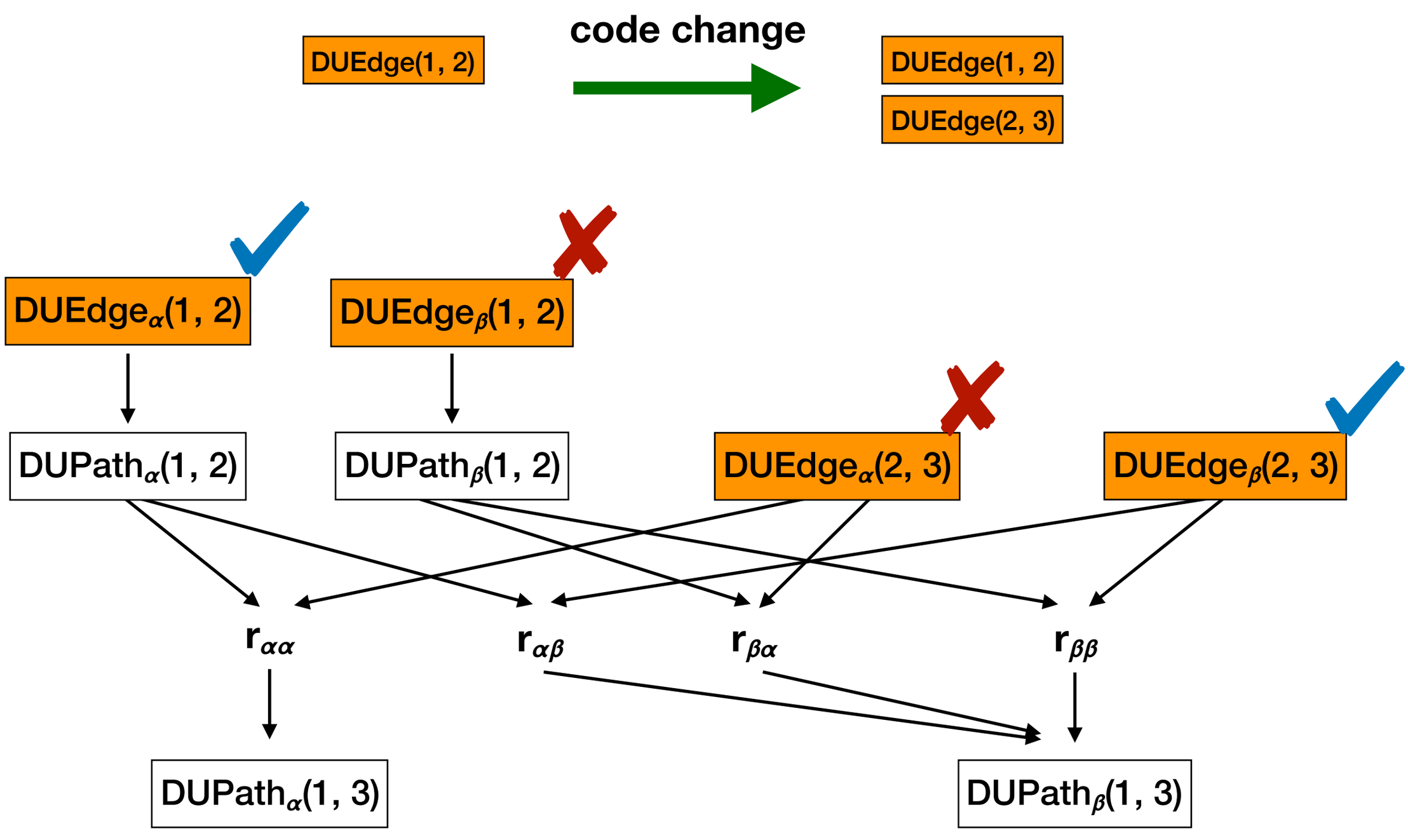
Example



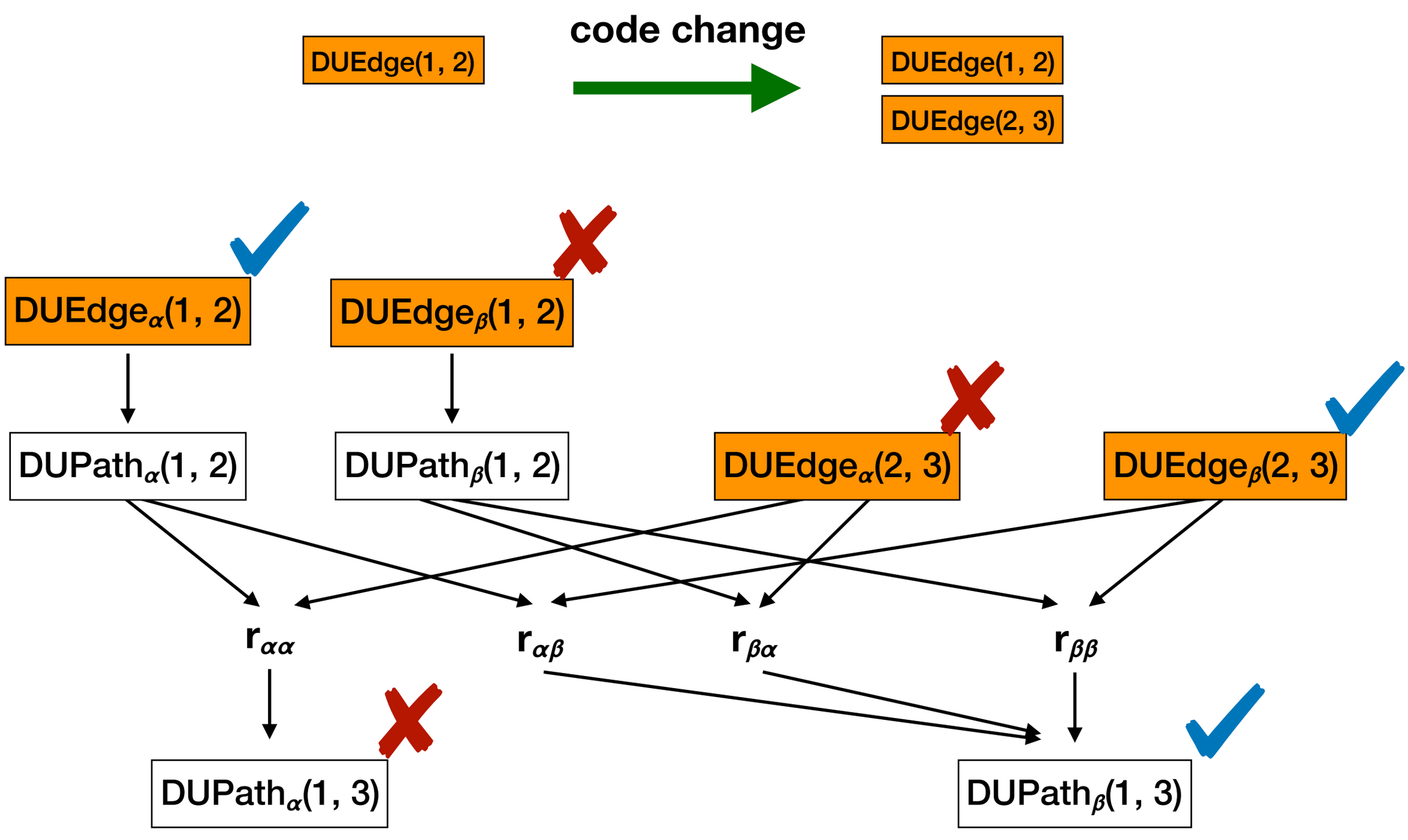
Example


DUEdge(1, 2)
code change
DUEdge(1, 2)
DUEdge(2, 3)
DUEdge$_\alpha$(1, 2)
DUEdge$_\beta$(1, 2)
DUPath$_\alpha$(1, 2)
DUPath$_\beta$(1, 2)
DUEdge$_\alpha$(2, 3)
DUEdge$_\beta$(2, 3)
r$_{\alpha\alpha}$
r$_{\alpha\beta}$
r$_{\beta\alpha}$
r$_{\beta\beta}$
DUPath$_\alpha$(1, 3)
DUPath$_\beta$(1, 3)

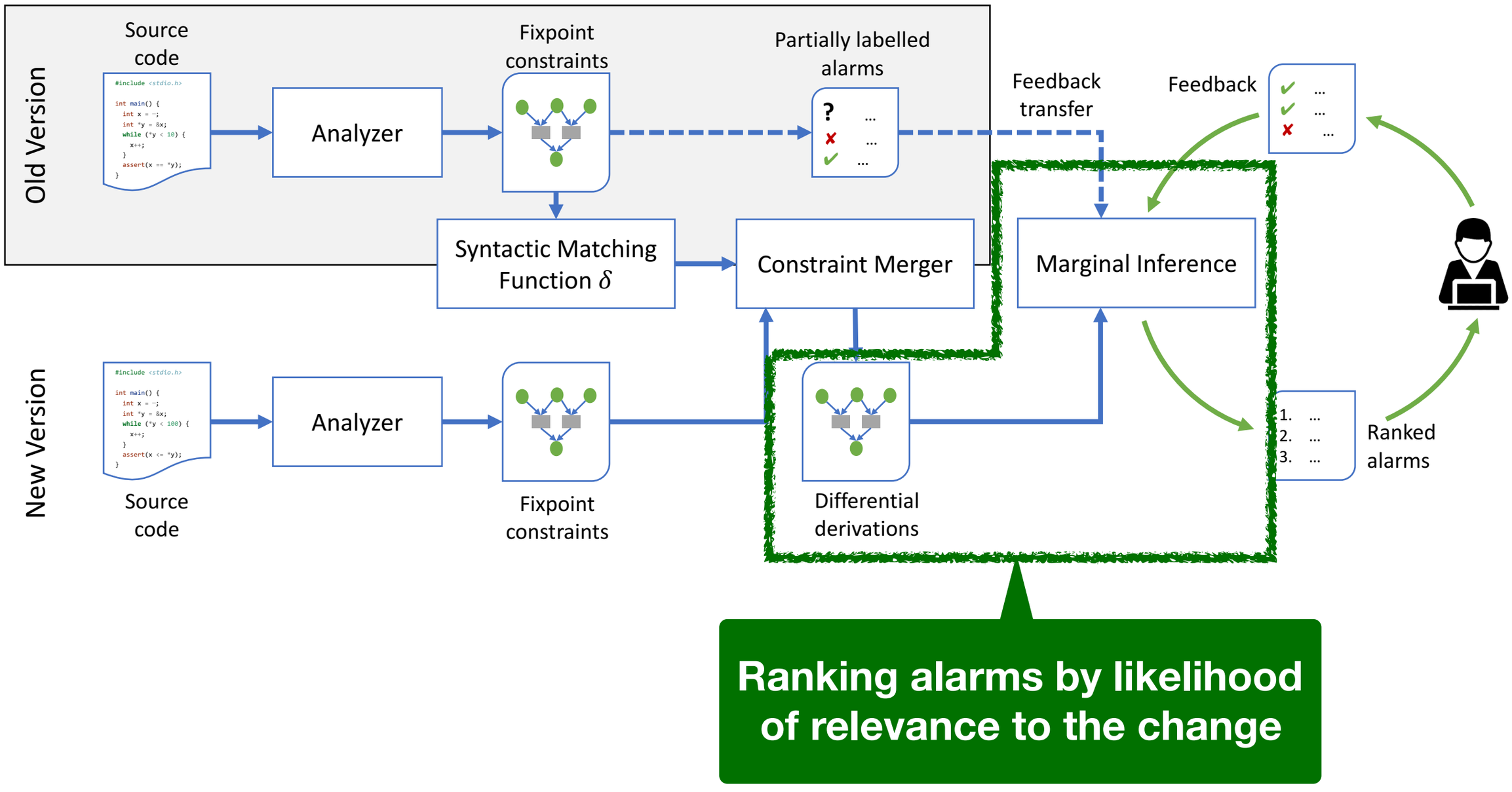
Example


DUEdge(1, 2)
code change
DUEdge(1, 2)
DUEdge(2, 3)
DUEdge$_\alpha$(1, 2)
DUEdge$_\beta$(1, 2)
DUPath$_\alpha$(1, 2)
DUPath$_\beta$(1, 2)
DUEdge$_\alpha$(2, 3)
DUEdge$_\beta$(2, 3)
r$_{\alpha\alpha}$
r$_{\alpha\beta}$
r$_{\beta\alpha}$
r$_{\beta\beta}$
DUPath$_\alpha$(1, 3)
DUPath$_\beta$(1, 3)

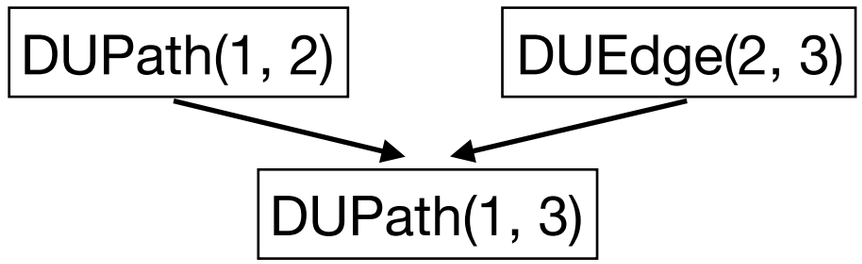
Example


DUEdge(1, 2)
code change
DUEdge(1, 2)
DUEdge(2, 3)
DUEdge$_\alpha$(1, 2)
DUEdge$_\beta$(1, 2)
DUPath$_\alpha$(1, 2)
DUPath$_\beta$(1, 2)
DUEdge$_\alpha$(2, 3)
DUEdge$_\beta$(2, 3)
r$_{\alpha\alpha}$
r$_{\alpha\beta}$
r$_{\beta\alpha}$
r$_{\beta\beta}$
DUPath$_\alpha$(1, 3)
DUPath$_\beta$(1, 3)

System Architecture



Bayesian Network



Logical Rule

Input relations

DUEdge(c1, c2) : Immediate data flow c1 to c2
Src(c) : Origin of potentially erroneous traces
Dst(c) : Potential program crash point

Output relations

DUPath(c1, c2) : Transitive data flow from c1 to c2
Alarm(c) : Potentially erroneous trace reaching c

Analysis Rules

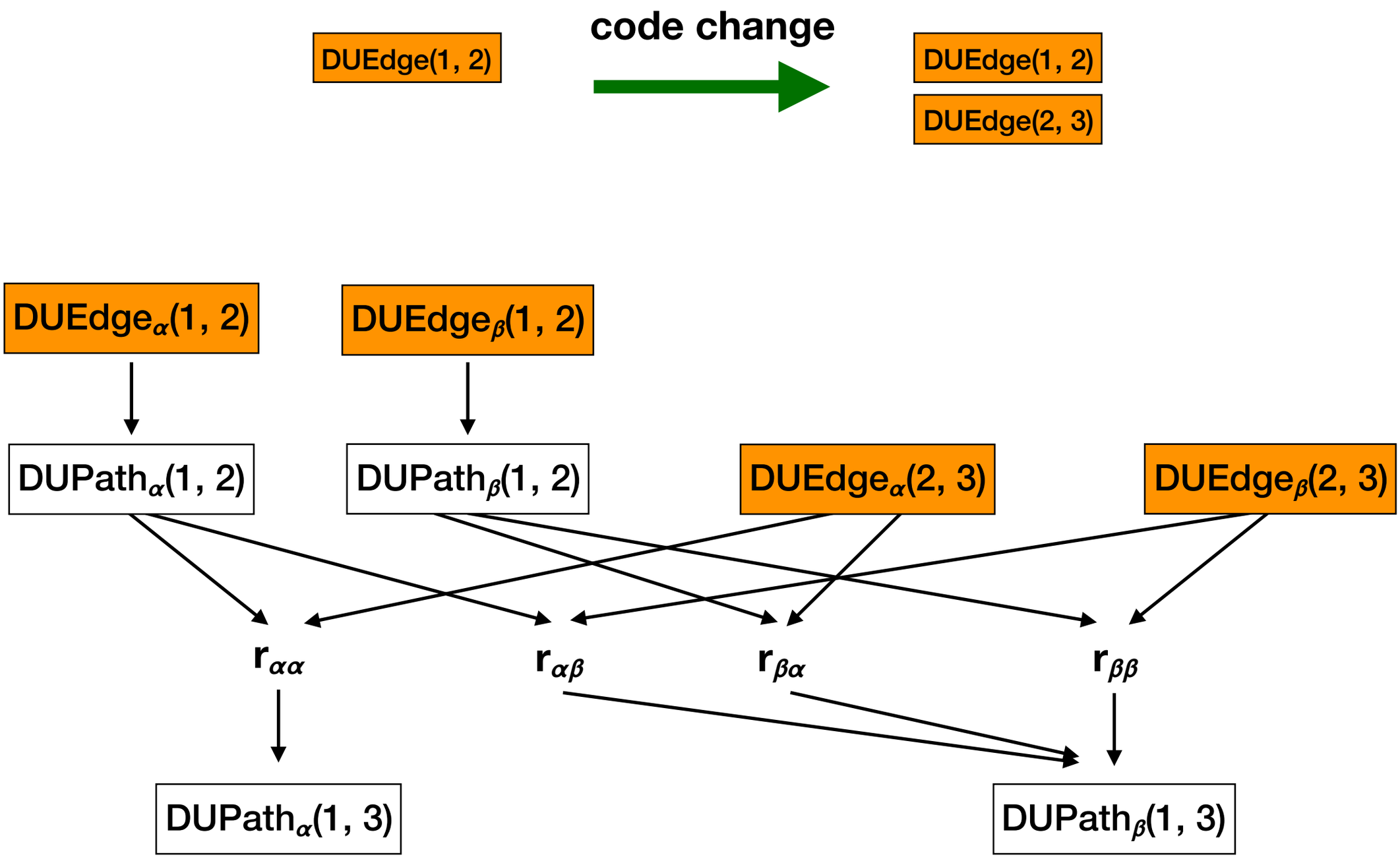
r1 : DUPath(c1, c2) :- DUEdge(c1, c2).
r2 : DUPath(c1, c3) :- DUPath(c1, c2), DUEdge(c1, c2).
r3 : Alarm(c2) :- DUPath(c1, c2), Src(c1), Dst(c2).

Probabilistic Rule

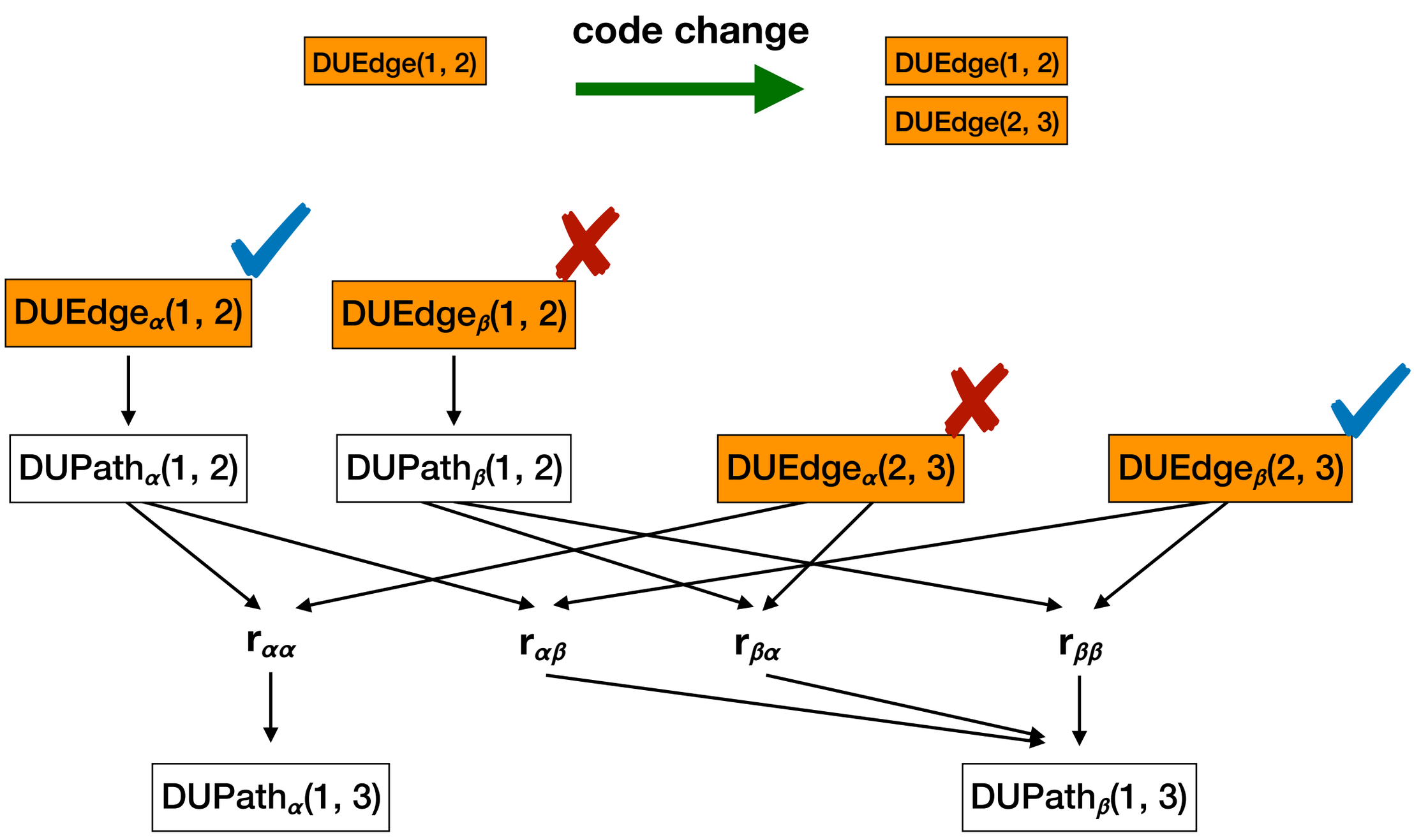
P(1,2)	E(2,3)	Pr(P(1,3) | H)
TRUE	TRUE	0.95*
TRUE	FALSE	0
FALSE	TRUE	0
FALSE	FALSE	0

*Prior probability is computed by an offline learning

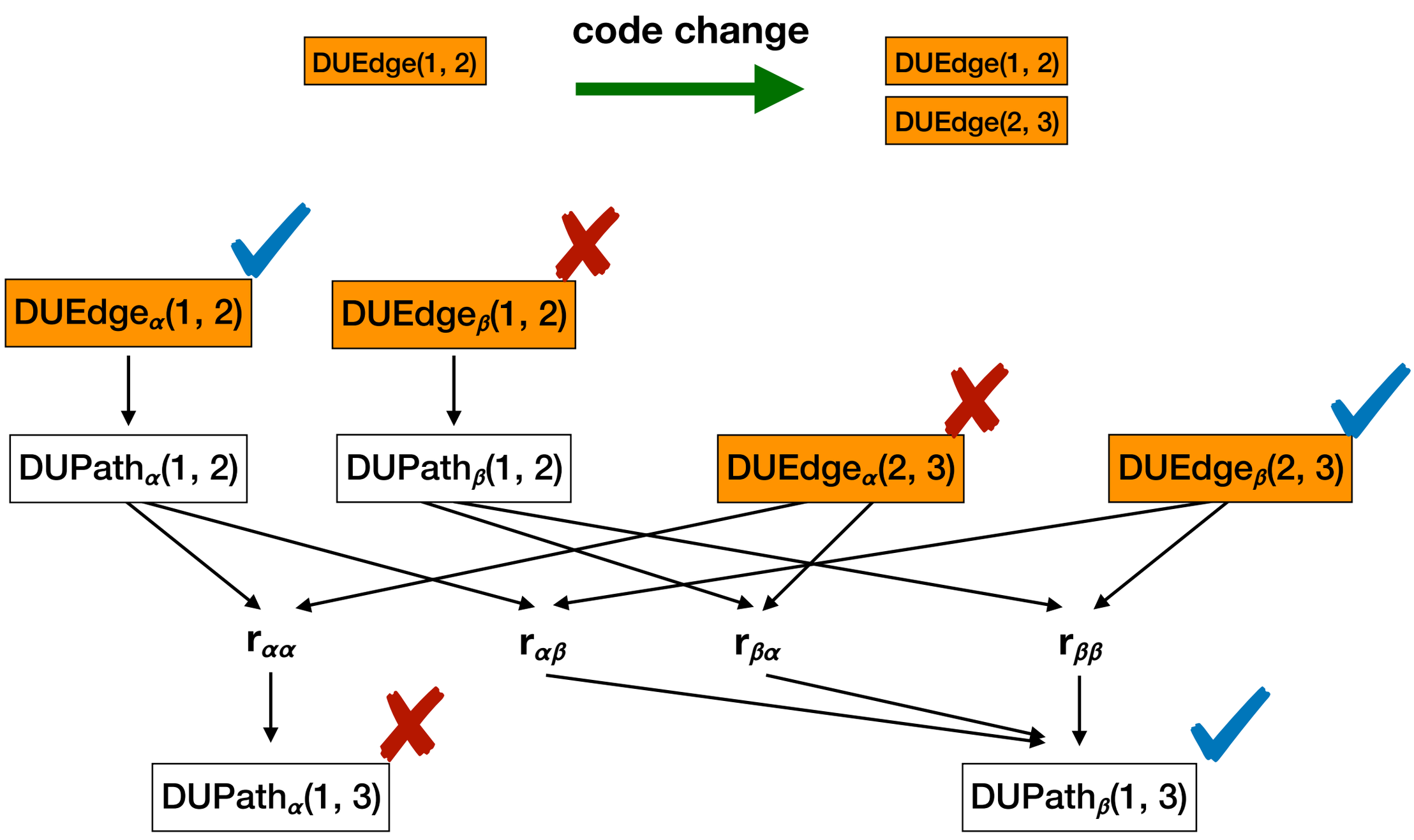
Bayesian Network



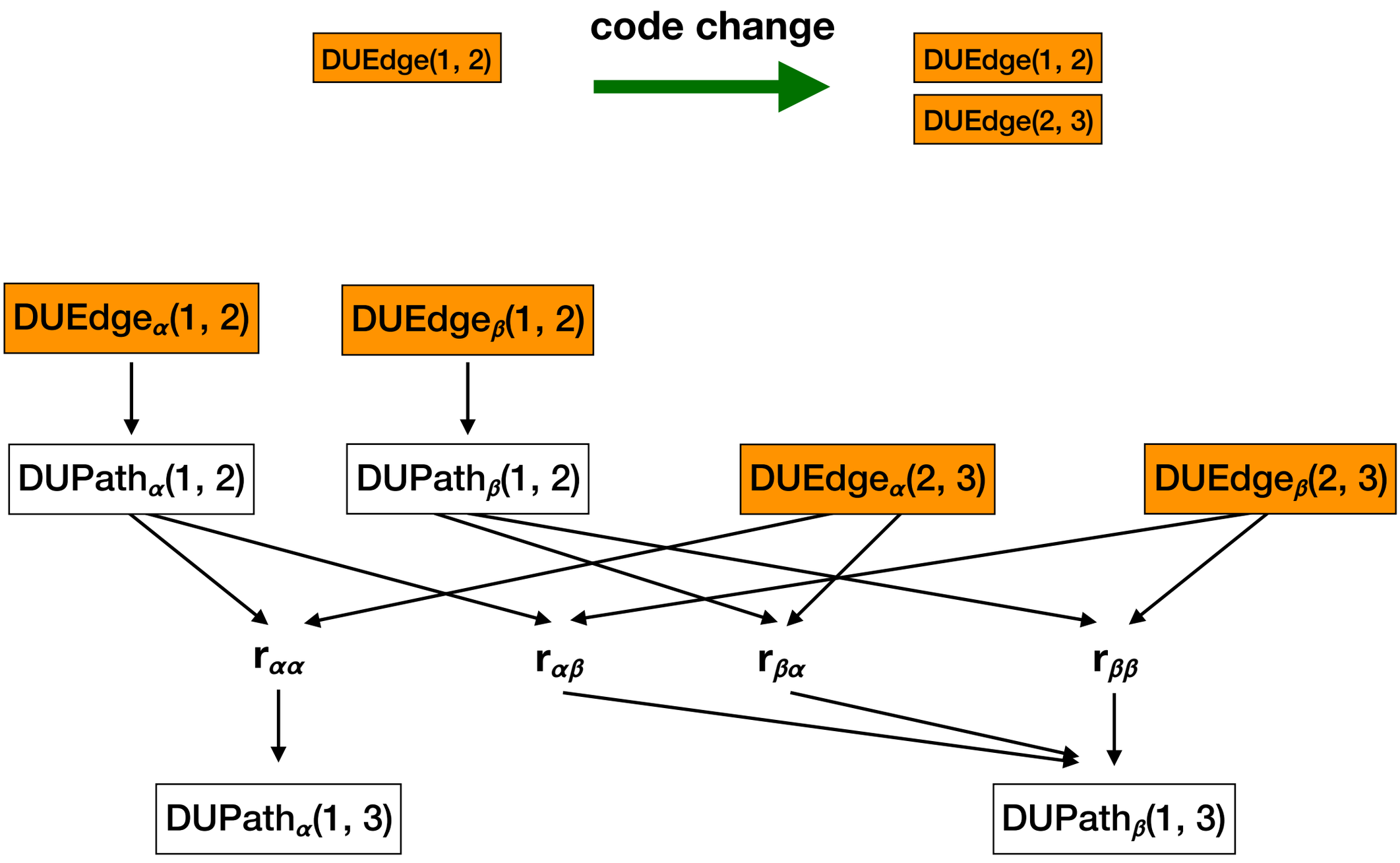
Bayesian Network



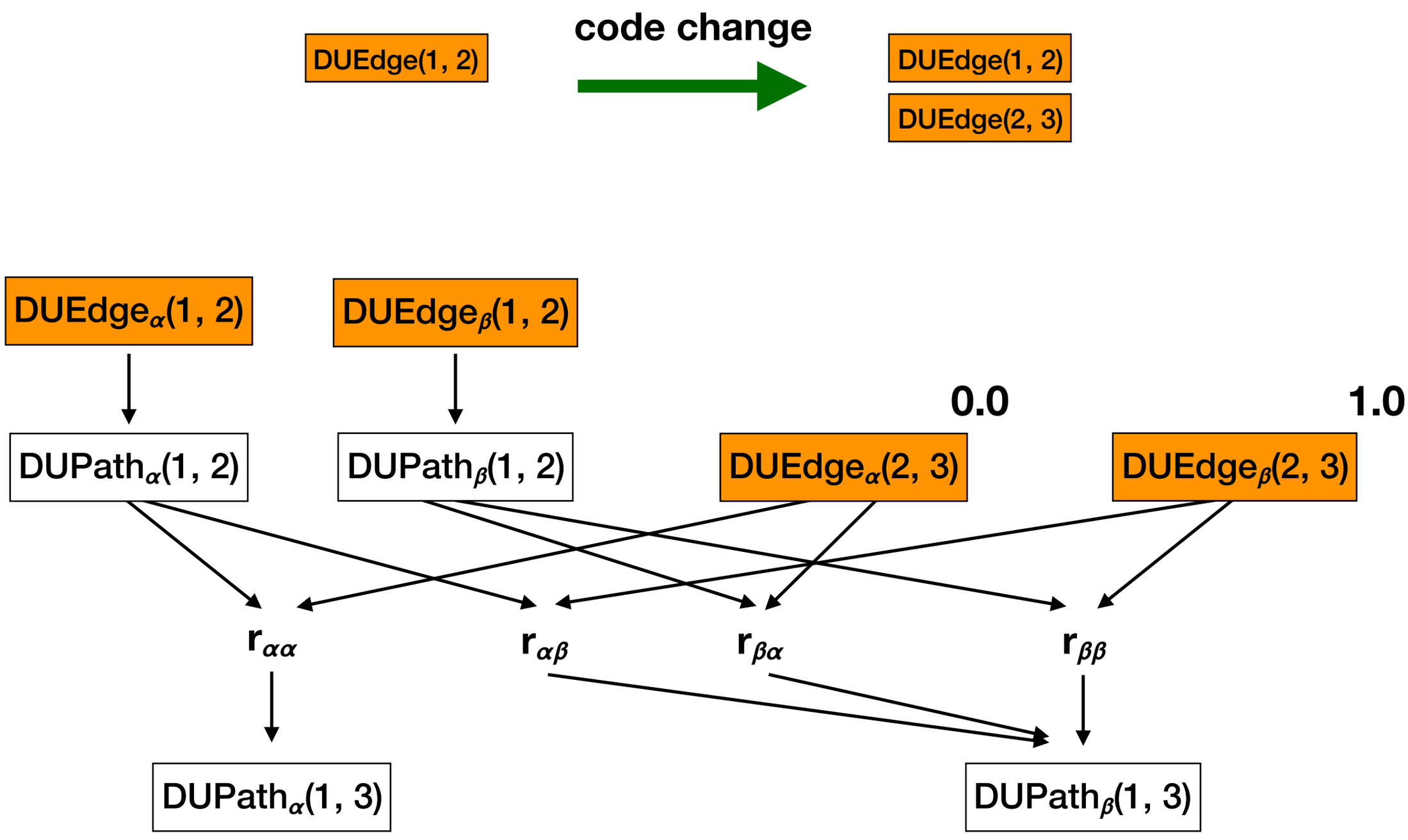
Bayesian Network



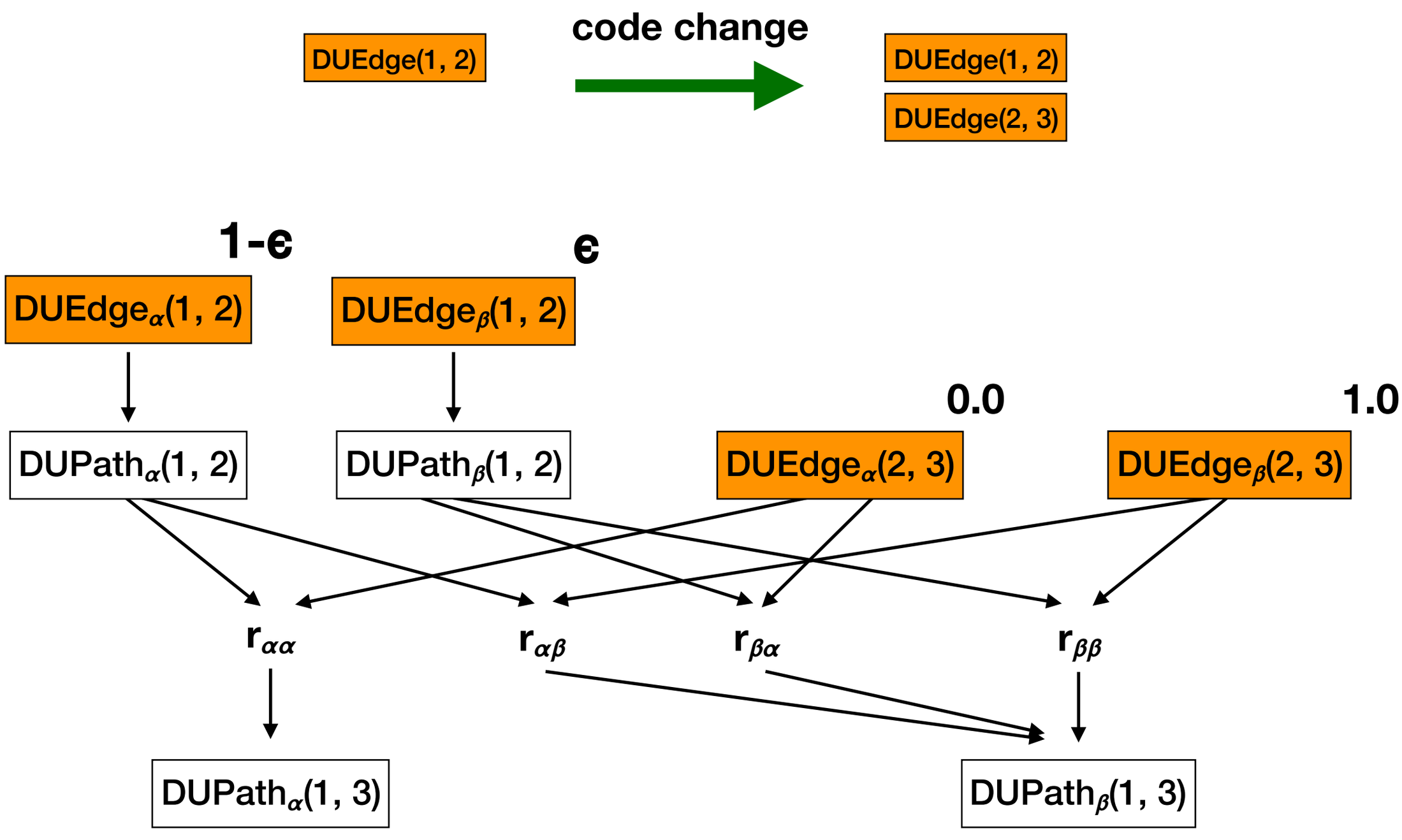
Bayesian Network



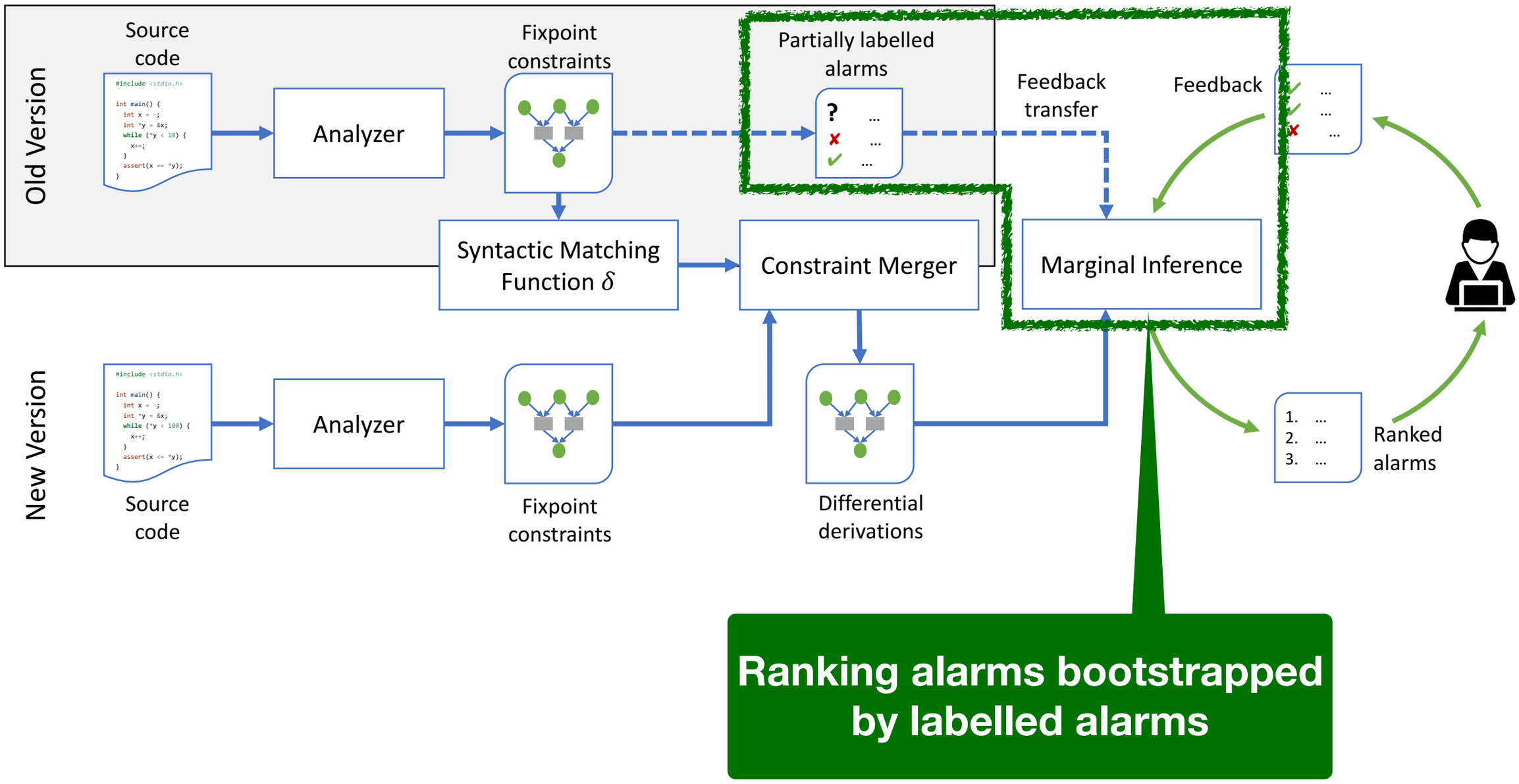
Bayesian Network



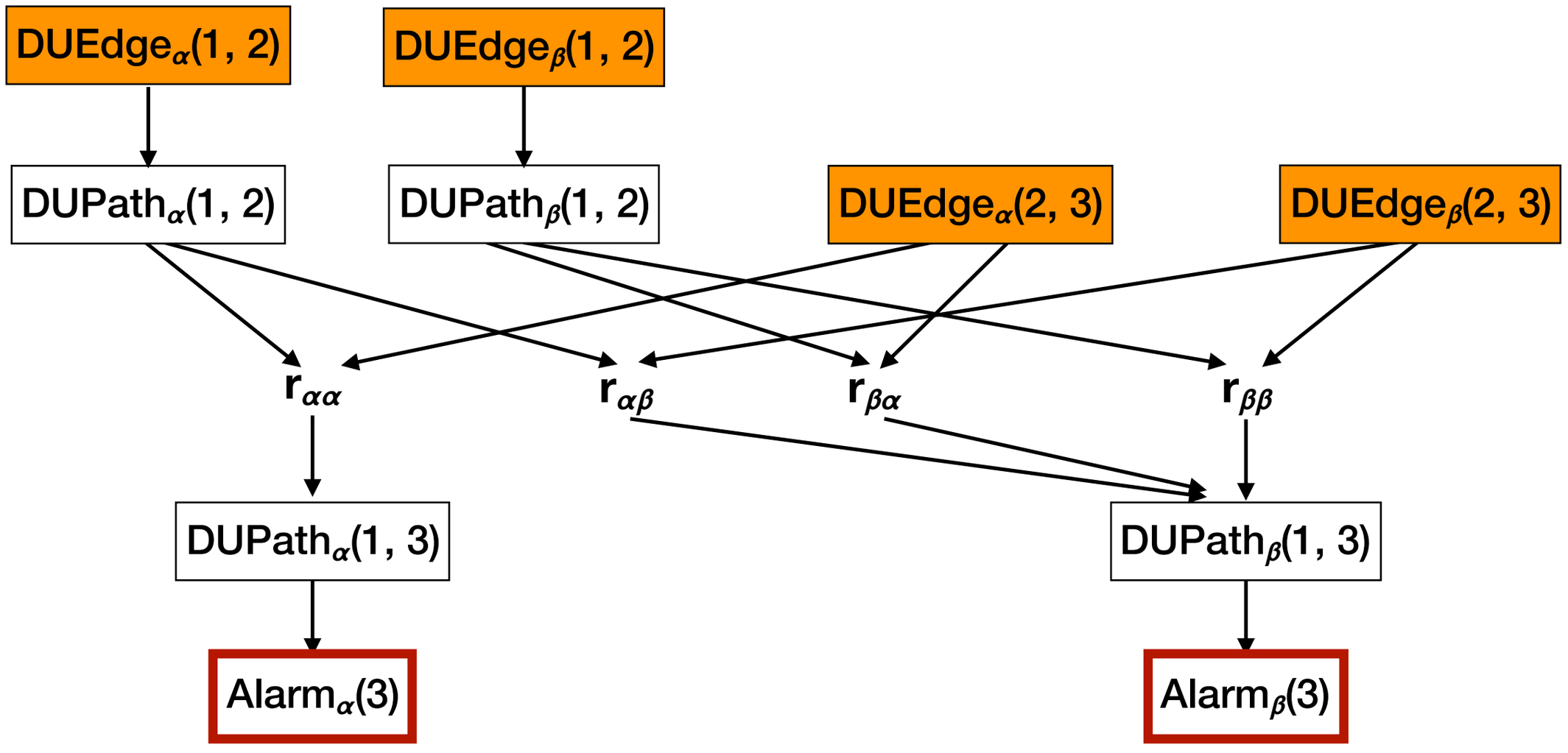
Bayesian Network



System Architecture

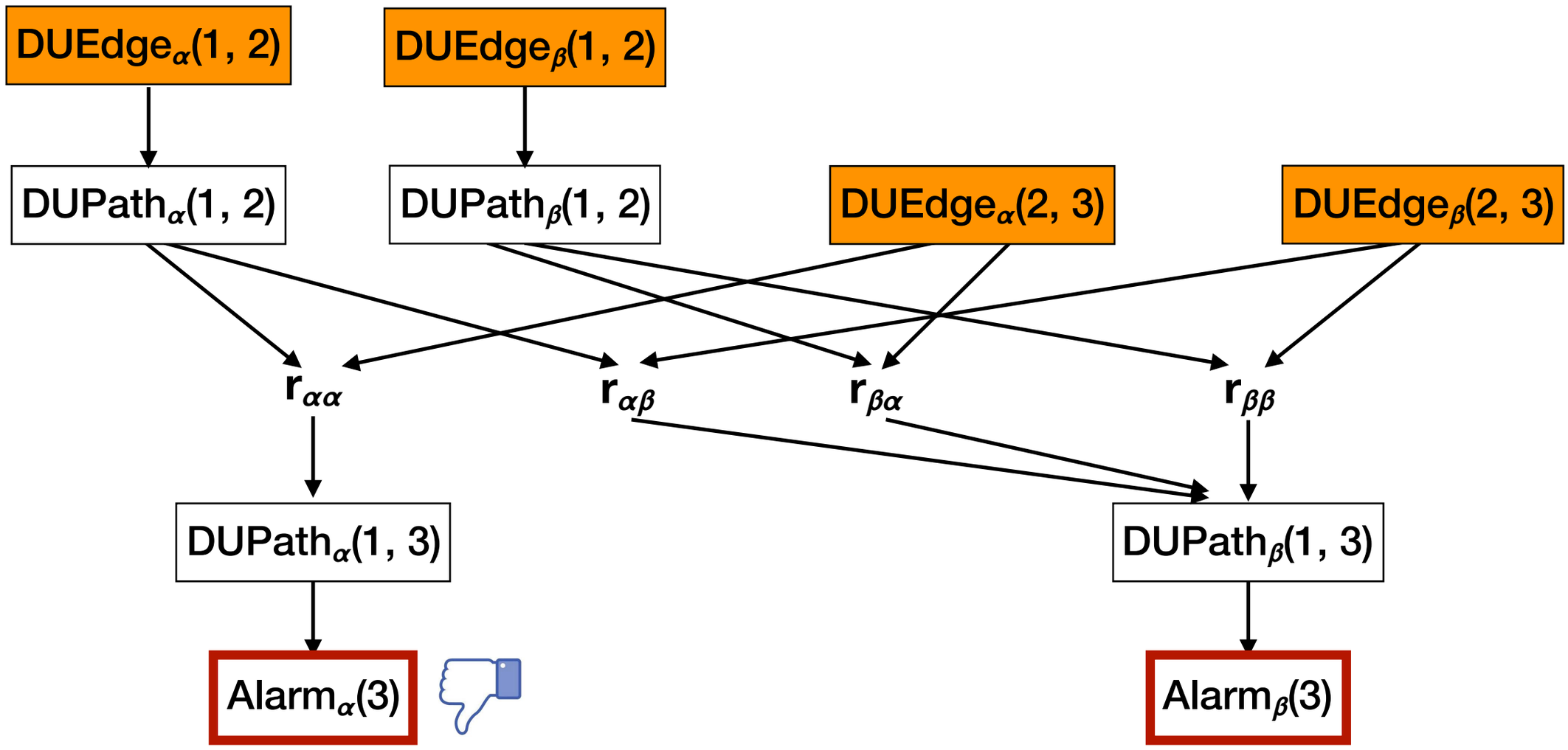


Feedback Transfer



Feedback Transfer

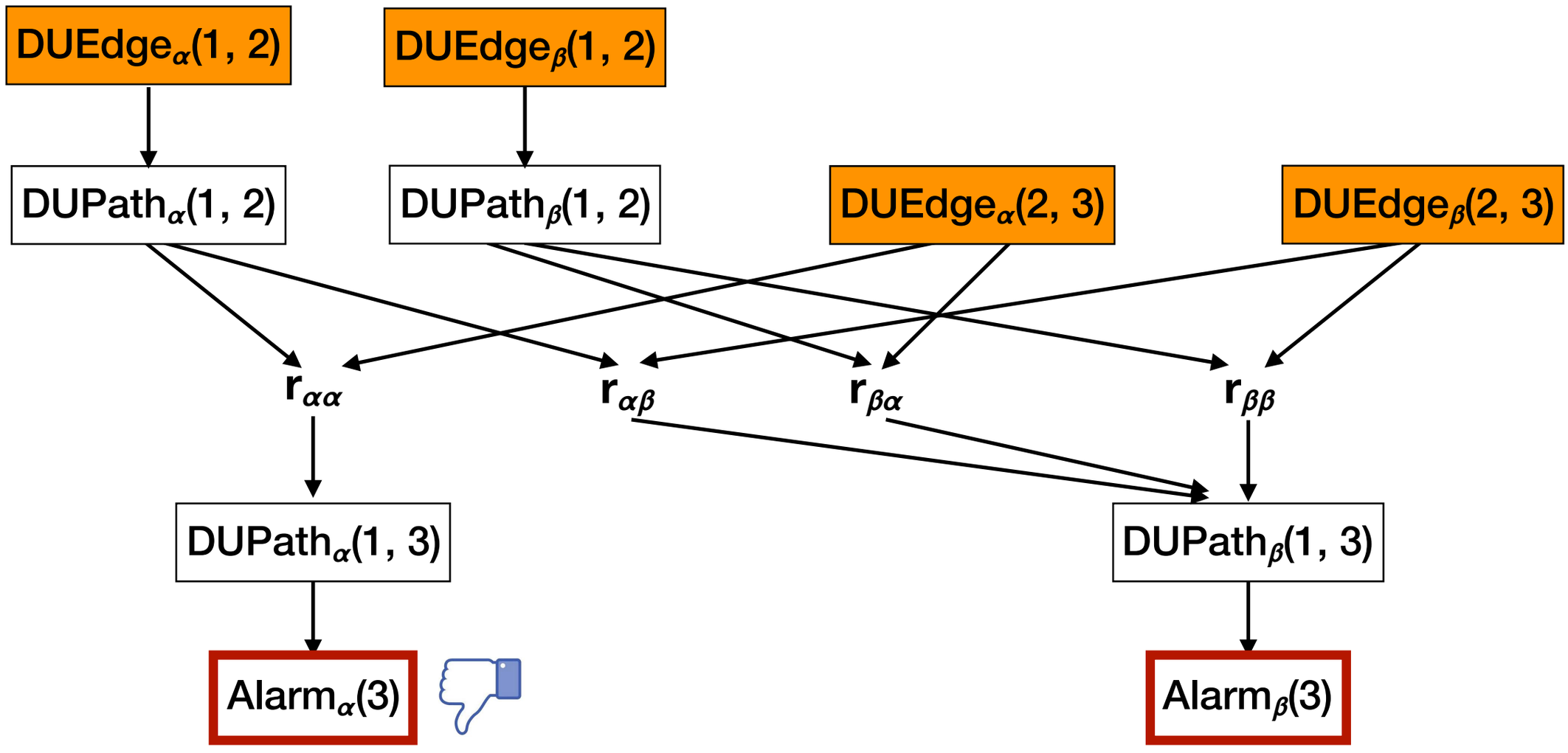
1. Conservative Mode



(if Alarm(3) was **FALSE** in the old version)

Feedback Transfer

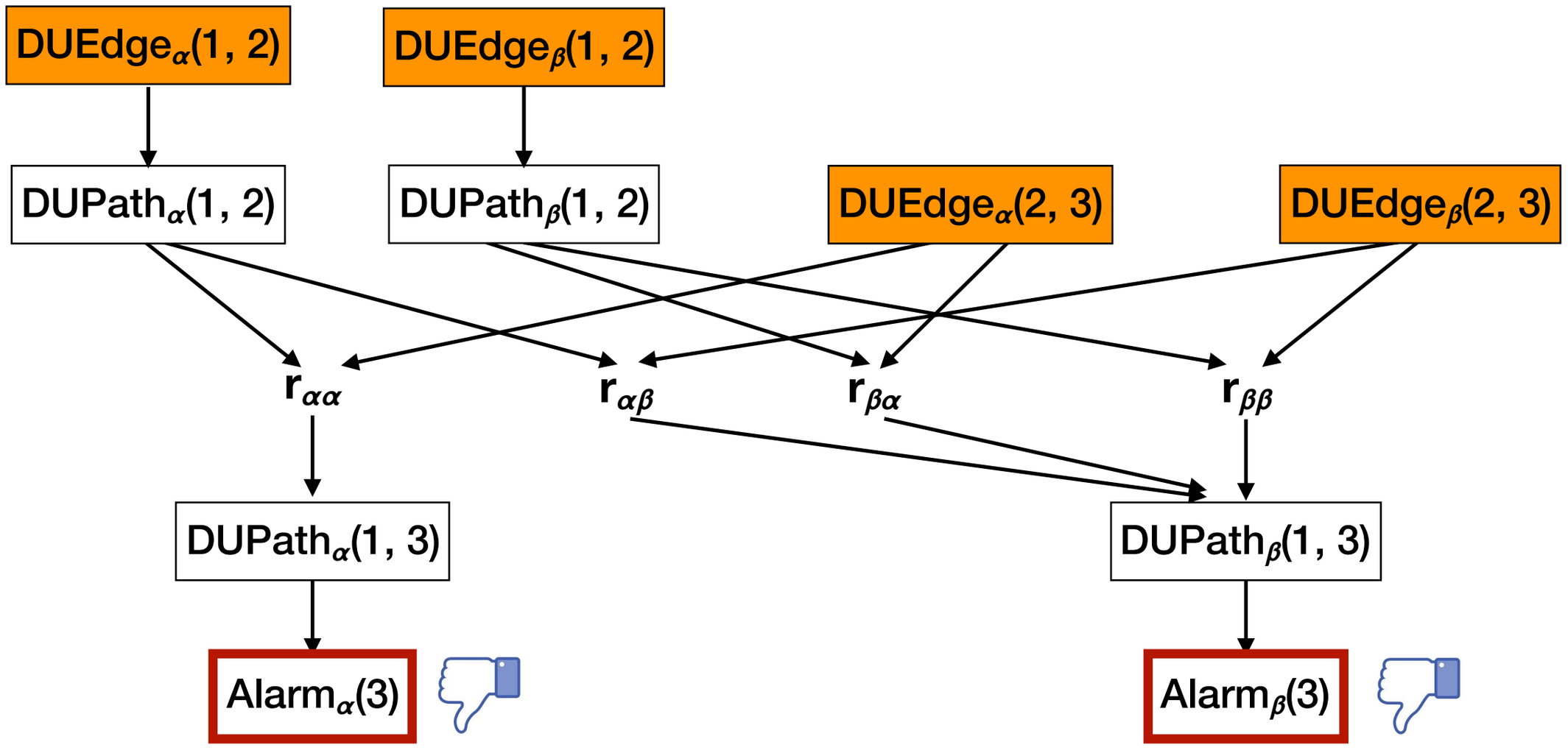
2. Strong Mode



(if Alarm(3) was **present** in the old version)

Feedback Transfer

3. Aggressive Mode



(if Alarm(3) was **present** in the old version)

(if Alarm(3) was **present** in the old version)

Benchmarks

Program	Version		Size (KLOC)		Diff	#Bugs	Bug Type
	Old	New	Old	New	(%)		
shntool	3.0.4	3.0.5	13	23	1	6	Int Overflow
latex2rtf	2.1.0	2.1.1	27	27	3	2	Format Str
urjtag	0.7	0.8	45	46	18	6	Format Str
optipng	0.5.2	0.5.3	60	61	2	1	Int Overflow
wget	1.11.4	1.12	42	65	47	6	Buf Overrun
readelf	2.23.2	2.24	63	65	6	1	Buf Overrun
grep	2.18	2.19	68	68	7	1	Buf Overrun
sed	4.2.2	4.3	48	83	40	1	Buf Overrun
sort	7.1	7.2	96	98	3	1	Buf Overrun
tar	1.27	1.28	108	112	4	1	Buf Overrun

Experimental Results

		Batch		Drake$_{Unsound}$				Drake$_{Sound}$		
Program	#Bugs	Old	New	#Misse	Init	FB	#Iter	Init	FB	#Iter
shntool	6	20	23	3	N/A	N/A	N/A	8	21	19
latex2rtf	2	7	13	0	5	6	5	12	9	6
urjtag	6	15	35	0	25	16	18	28	25	21
optipng	1	50	67	0	11	5	4	26	5	9
wget	6	850	792	0	122	139	54	392	317	122
readelf	1	841	882	0	28	4	4	216	182	25
grep	1	916	913	1	N/A	N/A	N/A	15	10	9
sed	1	572	818	0	262	209	60	154	118	41
sort	1	684	715	0	14	14	10	33	9	13
tar	1	1,229	1,369	0	23	29	15	56	82	32
Total	26	5,184	5,627	4	490	422	170	940	778	297

Experimental Results

		Batch		Drake$_{Unsound}$				Drake$_{Sound}$		
Program	#Bugs	Old	New	#Misse	Init	FB	#Iter	Init	FB	#Iter
shntool	6	20	23	3	N/A	N/A	N/A	8	21	19
latex2rtf	2	7	13	0	5	6	5	12	9	6
urjtag	6	15	35	0	25	16	18	28	25	21
optipng	1	50	67	0	11	5	4	26	5	9
wget	6	850	792	0	122	139	54	392	317	122
readelf	1	841	882	0	28	4	4	216	182	25
grep	1	916	913	1	N/A	N/A	N/A	15	10	9
sed	1			0	262	209	60	154	118	41
sort	1			0	14	14	10	33	9	13
tar	1	1,229	1,369	0	23	29	15	56	82	32
Total	26	5,184	5,627	4	490	422	170	940	778	297

 # Alarms of batch-mode analyses

Experimental Results

		Batch		Drake$_{Unsound}$				Drake$_{Sound}$		
Program	#Bugs	Old	New	#Misse	Init	FB	#Iter	Init	FB	#Iter
shntool	6	20	23	3	N/A	N/A	N/A	8	21	19
latex2rtf	2	7	13	0	5	6	5	12	9	6
urjtag	6	15	35	0	25	16	18	28	25	21
optipng	1	50	67	0	11	5	4	26	5	9
wget	6	850	792	0	122	139	54	392	317	122
readelf	1	841	882	0	28	4	4	216	182	25
grep	1	816	913	1	N/A	N/A	N/A	15	10	9
sed	1	1,153	1,293	0	262	209	60	154	118	41
sort	1	1,153	1,293	0	14	14	10	33	9	13
tar	1	1,229	1,369	0	23	29	15	56	82	32
Total	26	5,184	5,627	4	490	422	170	940	778	297

Alarms of batch-mode analyses

Experimental Results

		Batch		Drake$_{Unsound}$				Drake$_{Sound}$		
Program	#Bugs	Old	New	#Misse	Init	FB	#Iter	Init	FB	#Iter
shntool	6	20	23	3	N/A	N/A	N/A	8	21	19
latex2rtf	2	7	13	0	5	6	5	12	9	6
urjtag	6	15	35	0	25	16	18	28	25	21
optipng	1	50	67	0	11	5	4	26	5	9
wget	6	850	792	0	122	139	54	392	317	122
readelf	1	841	882	0	28	4	4	216	182	25
grep	1	916	913	1	N/A	N/A	N/A	15	10	9
sed	1			0				118		41
sort	1			0				9		13
tar	1	1,229	1,369	0	23	29	15	56	82	32
Total	26	5,184	5,627	4	490	422	170	940	778	297

 # Alarms of batch-mode analyses

Ranking by likelihood of relevance to the change

Experimental Results

Program	#Bugs	Batch Old	Batch New	Drake$_{Unsound}$ #Misse	Drake$_{Unsound}$ Init	Drake$_{Unsound}$ FB	Drake$_{Unsound}$ #Iter	Drake$_{Sound}$ Init	Drake$_{Sound}$ FB	Drake$_{Sound}$ #Iter
shntool	6	20	23	3	N/A	N/A	N/A	8	21	19
latex2rtf	2	7	13	0	5	6	5	12	9	6
urjtag	6	15	35	0	25	16	18	28	25	21
optipng	1	50	67	0	11	5	4	26	5	9
wget	6	850	792	0	122	139	54	392	317	122
readelf	1	841	882	0	28	4	4	216	182	25
grep	1	916	913	1	N/A	N/A	N/A	15	10	9
sed	1			0	262					41
sort	1			0	14					13
tar	1	1,229	1,369	0	23	29	15	56	82	32
Total	26	5,184	5,627	4	490	422	170	940	778	297

 # Alarms of batch-mode analyses

Ranking bootstrapped by labelled alarms

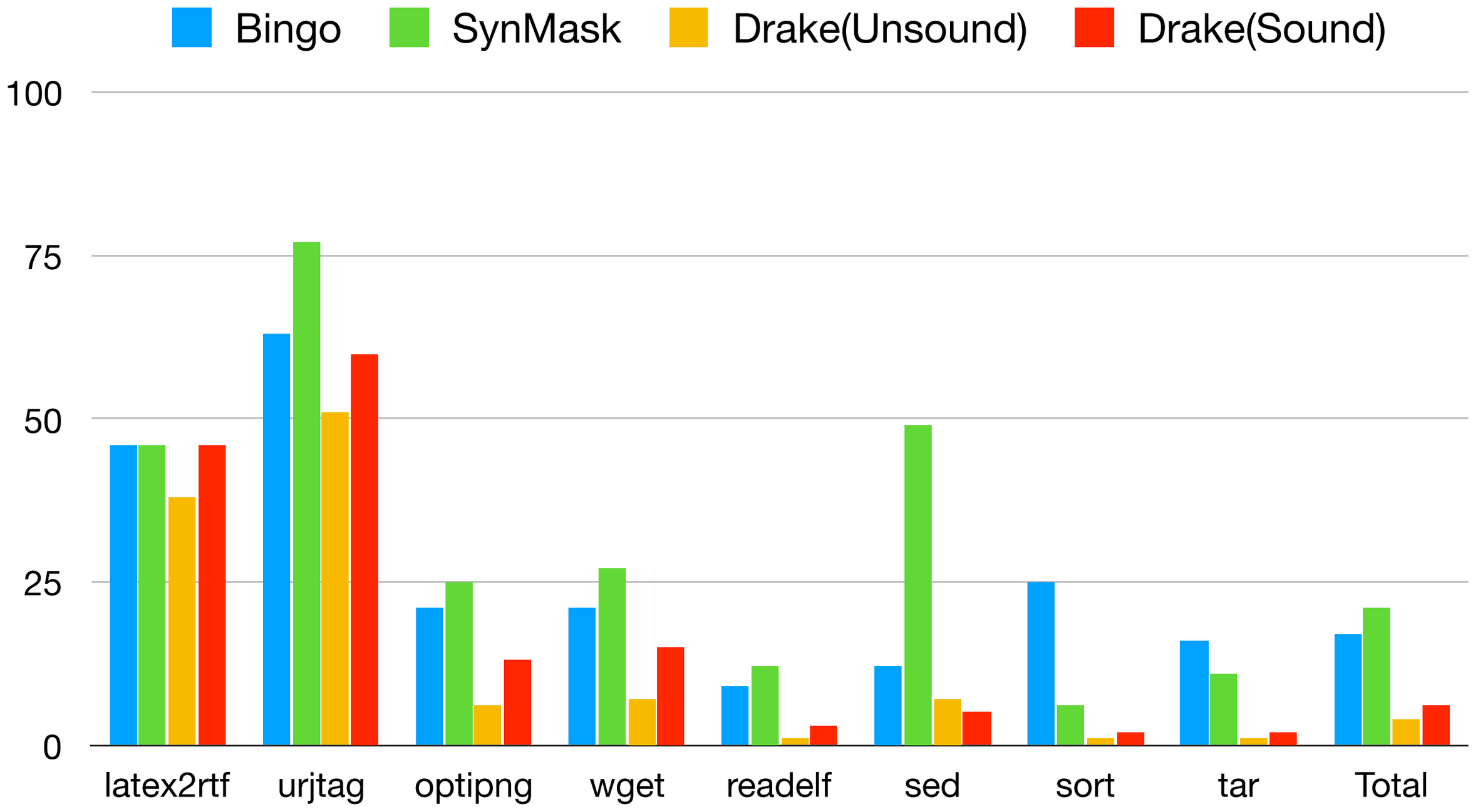
Experimental Results

Program	#Bugs	Batch Old	Batch New	Drake$_{Unsound}$ #Misse	Drake$_{Unsound}$ Init	Drake$_{Unsound}$ FB	Drake$_{Unsound}$ #Iter	Drake$_{Sound}$ Init	Drake$_{Sound}$ FB	Drake$_{Sound}$ #Iter
shntool	6	20	23	3	N/A	N/A	N/A	8	21	19
latex2rtf	2	7	13	0	5	6	5	12	9	6
urjtag	6	15	35	0	25	16	18	28	25	21
optipng	1	50	67	0	11	5	4	26	5	9
wget	6	850	792	0	122	139	54	392	317	122
readelf	1	841	882	0	28	4	4	216	182	25
grep	1	916	913	1	N/A	N/A	N/A	15	10	9
sed	1			0	262	209				
sort	1			0	14	14				
tar	1	1,229	1,369	0	23	29	15	56	82	32
Total	26	5,184	5,627	4	490	422	170	940	778	297

 # Alarms of batch-mode analyses

Ranking by user feedback

Experimental Results



Conclusion

- **AI-based** programming reasoning system
- **Interactive** and **continuous** reasoning via Bayesian Network
- Future work:
  - Github-scale system
  - packaging as library
  - holistic program reasoning system
    (static analysis with testing, patch, patterns, etc)