

Petrification

Software Model Checking for Programs with Dynamic Thread Management

Frank Schüssele Matthias Heizmann Dominik Klumpp Lars Nitzke

University of Freiburg, Germany

Practical verification algorithm for programs with dynamic thread management

Practical verification algorithm for programs with **dynamic thread management**

- used in C (pthread library)

- used in C (pthread library)
- fork (pthread_create) and join (pthread_join)

Dynamic thread management

- used in C (pthread library)
- fork (pthread_create) and join (pthread_join)
- depends on control flow and data flow of the program

Example

main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

Example

main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

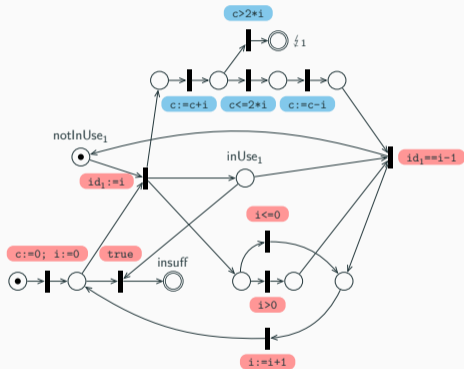
worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

Thread width

Maximum number of instances of a thread template
at any point in time
in any execution

Petrification



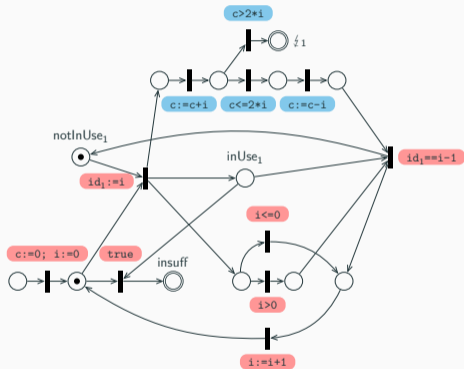
main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

Petrification



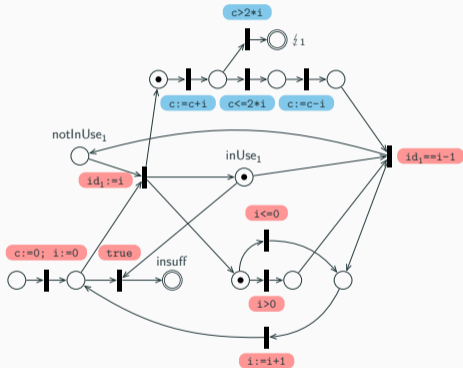
main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

Petrification



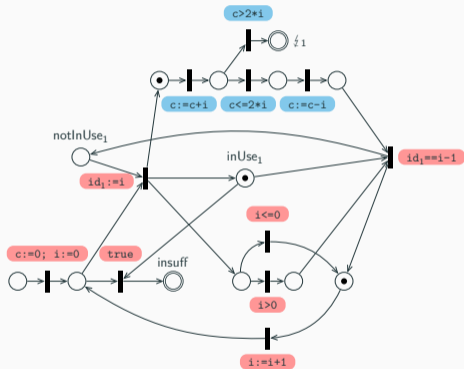
main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

Petrification



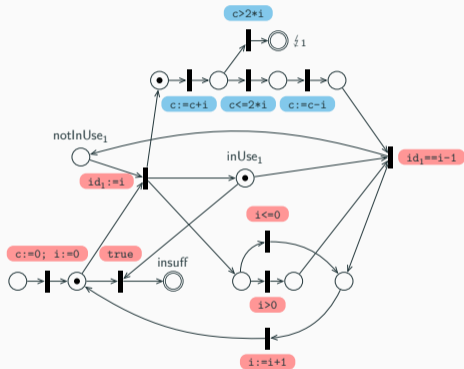
main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

Petrification



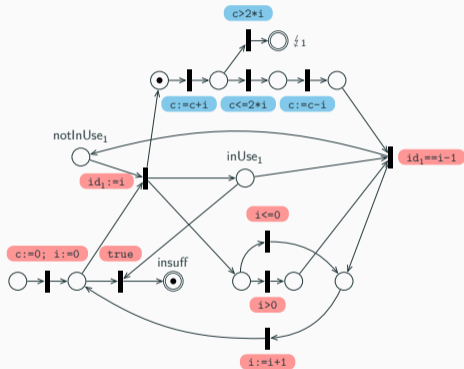
main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

Petrification



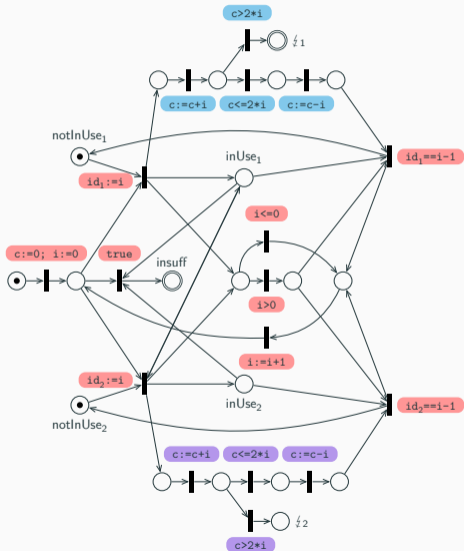
main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

Petrification



main

```
c := 0; i := 0; // global
while (true) {
  fork i worker();
  if (i > 0) {
    join i - 1;
  }
  i := i + 1;
}
```

worker

```
c := c + i;
assert c <= 2 * i;
c := c - i;
```

- fork: no thread instance is used more than once

- fork: no thread instance is used more than once
- join: matching instance is destroyed, depends on evaluation of thread

- in ULTIMATE GEMCUTTER, ULTIMATE AUTOMIZER, ULTIMATE TAIPAN for verification of C programs
- top 3 sound tools in SV-COMP 2024 (in category ConcurrencySafety)

Practical verification algorithm for programs with dynamic thread management

Practical verification algorithm for programs with dynamic thread management

- complete for programs with a finite thread width