A Practical Algorithm for Structure Embedding

Charlie Murphy and Zak Kincaid

January 14, 2019

PRINCETON UNIVERSITY

Overview

1. Use in Multi-threaded Verification

2. Structure Embedding

3. MatchEmbeds

Cartesian Predicate Abstraction

```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```

Cartesian Predicate Abstraction

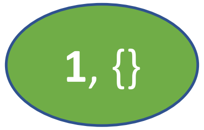
$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

```
main():
  x := *
  y := 0
  if (x < y)
     x := 0
  y := 2
  assert(x != 0)
  // rest of program
```

Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

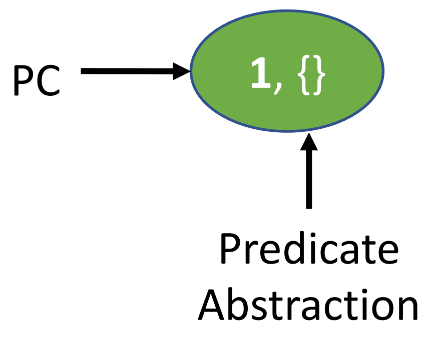
```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

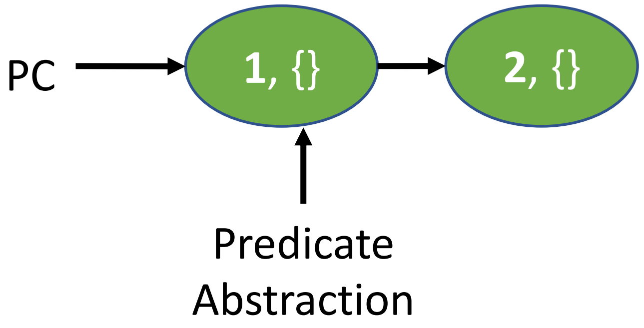
```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

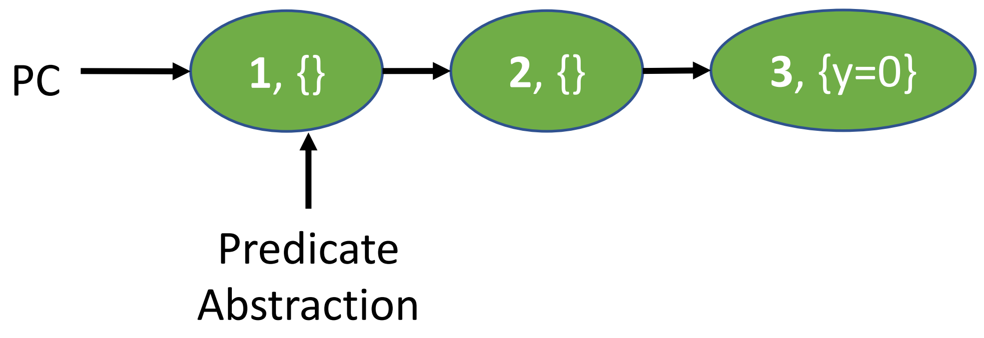
```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

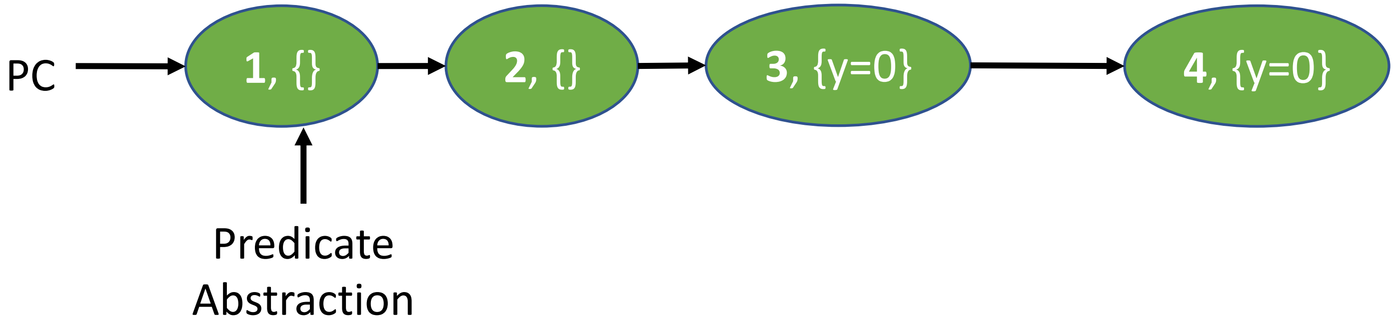
```
main():
  x := *
  y := 0
  if (x < y)
     x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

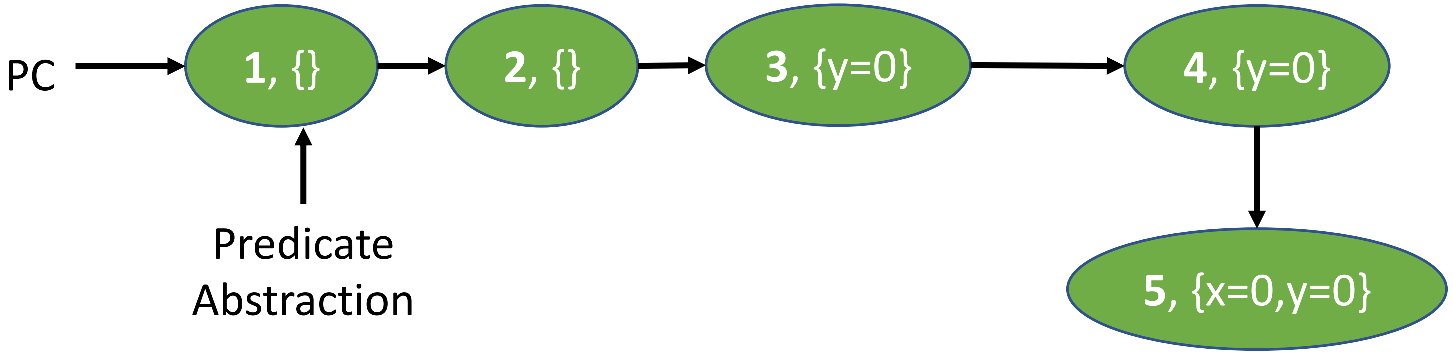
```
main():
  x := *
  y := 0
  if (x < y)
     x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

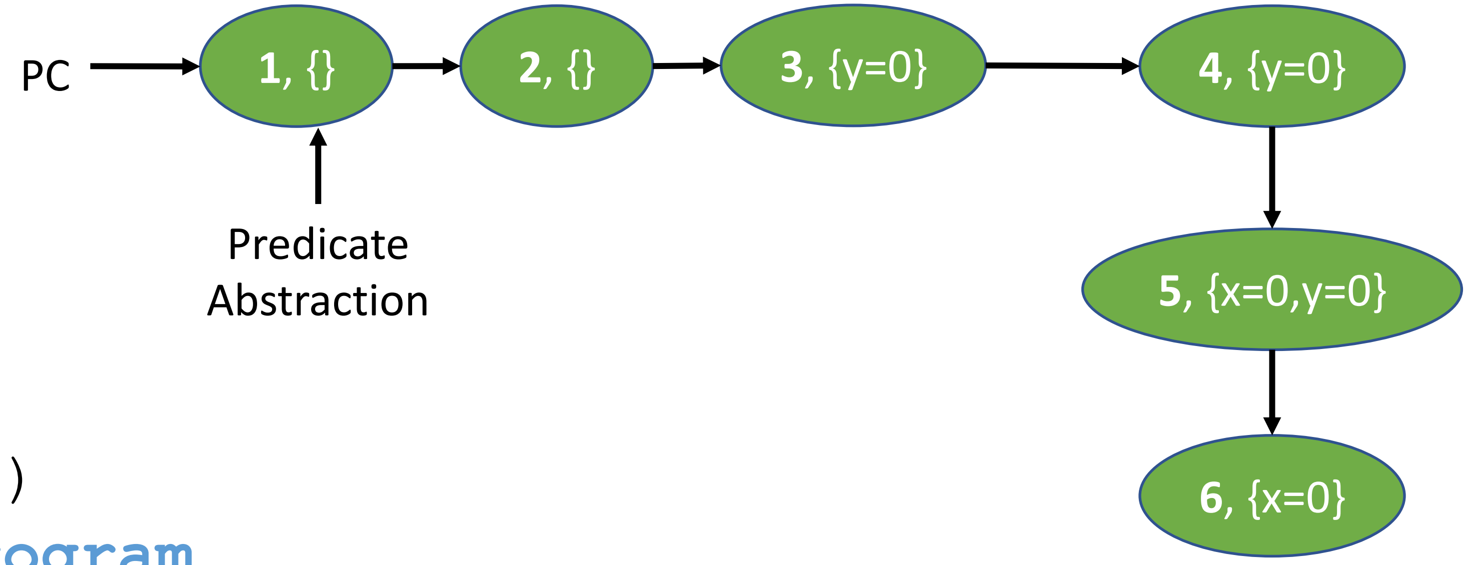
```
main():
  x := *
  y := 0
  if (x < y)
     x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

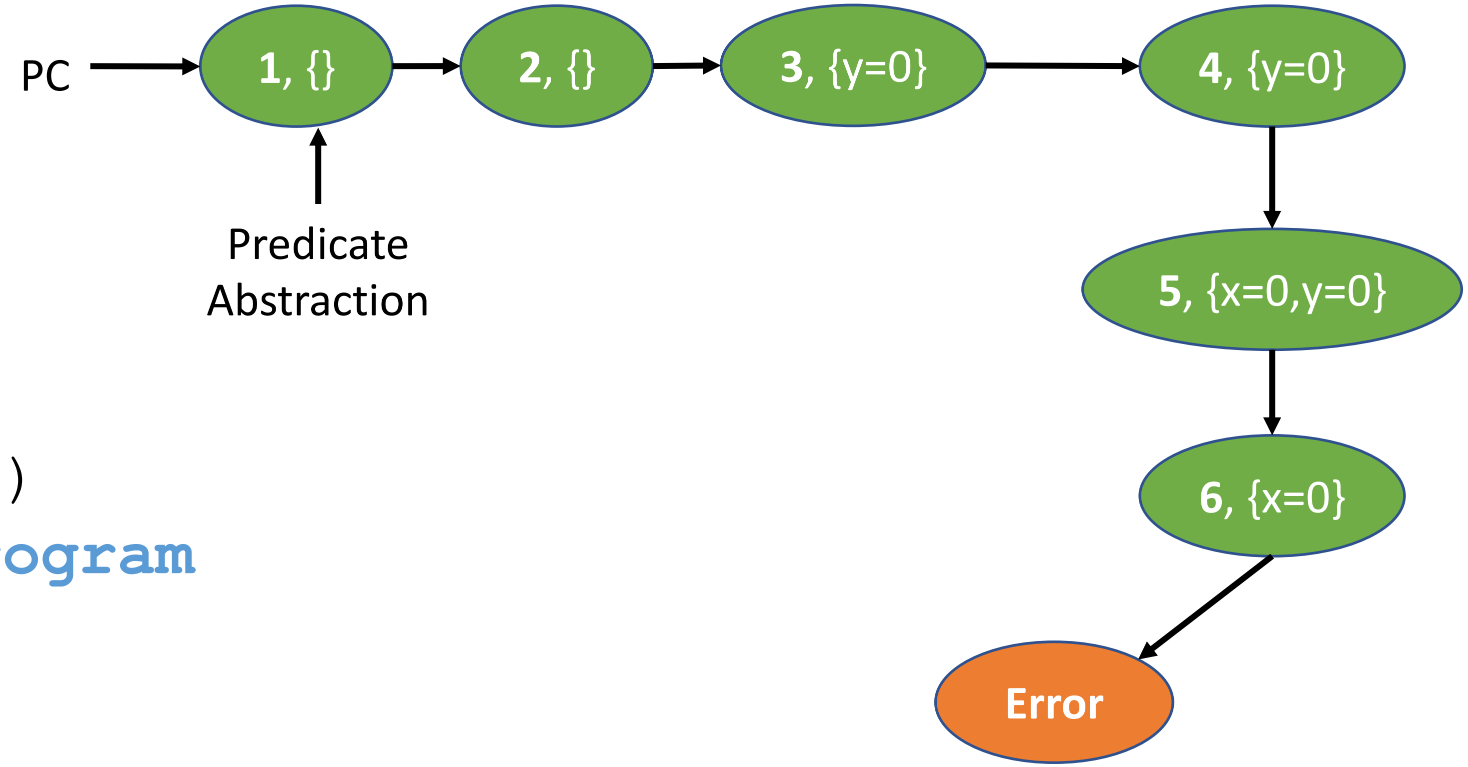
```
main():
  x := *
  y := 0
  if (x < y)
     x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

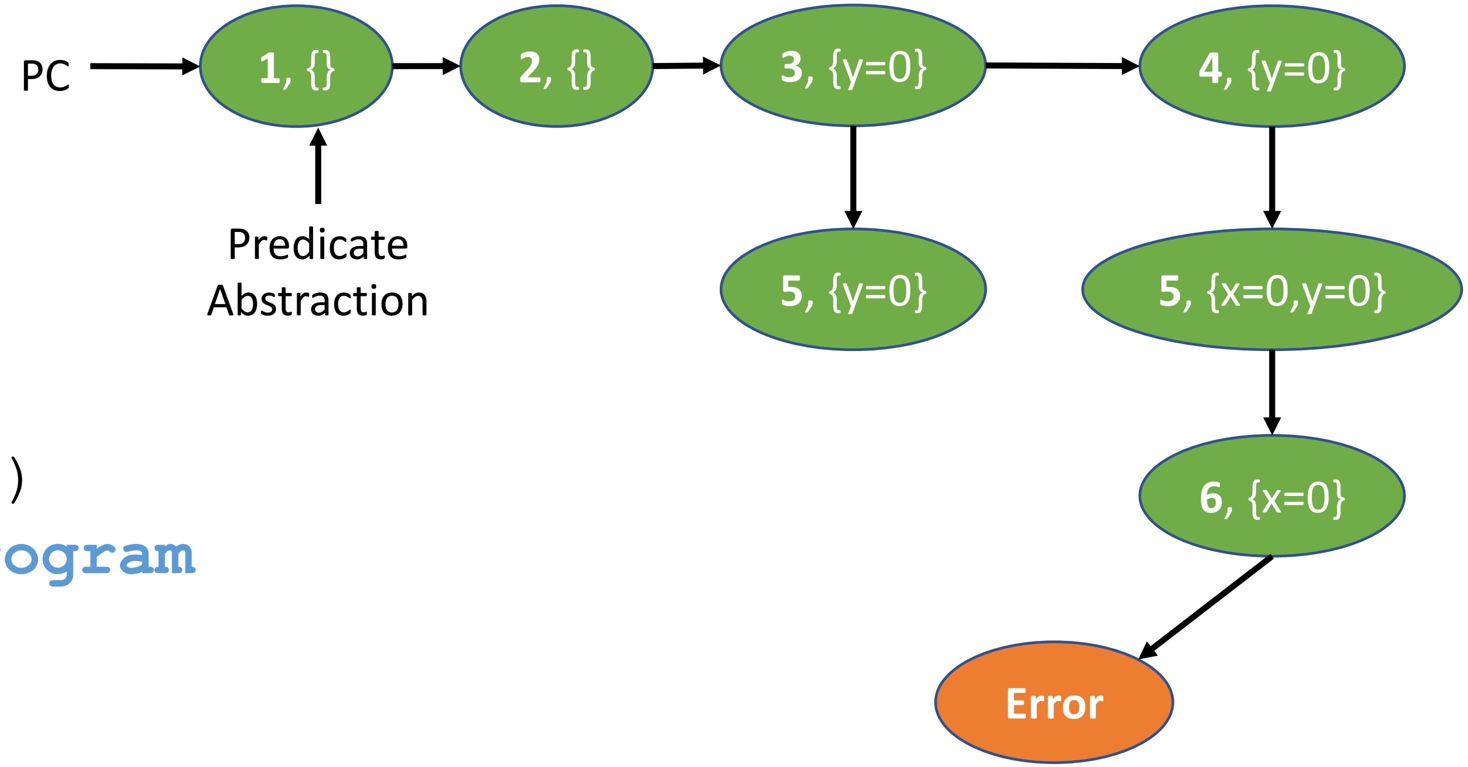
```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

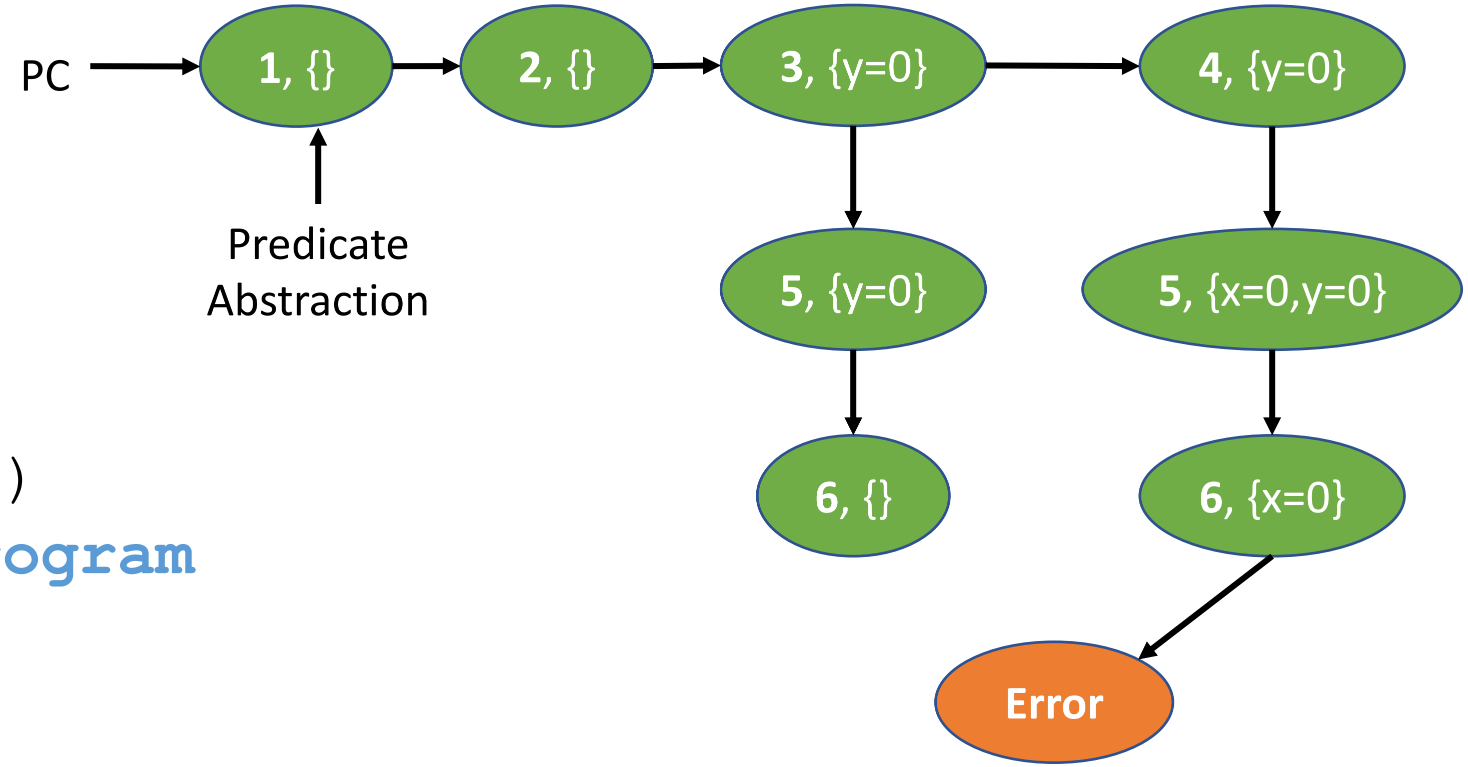
```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

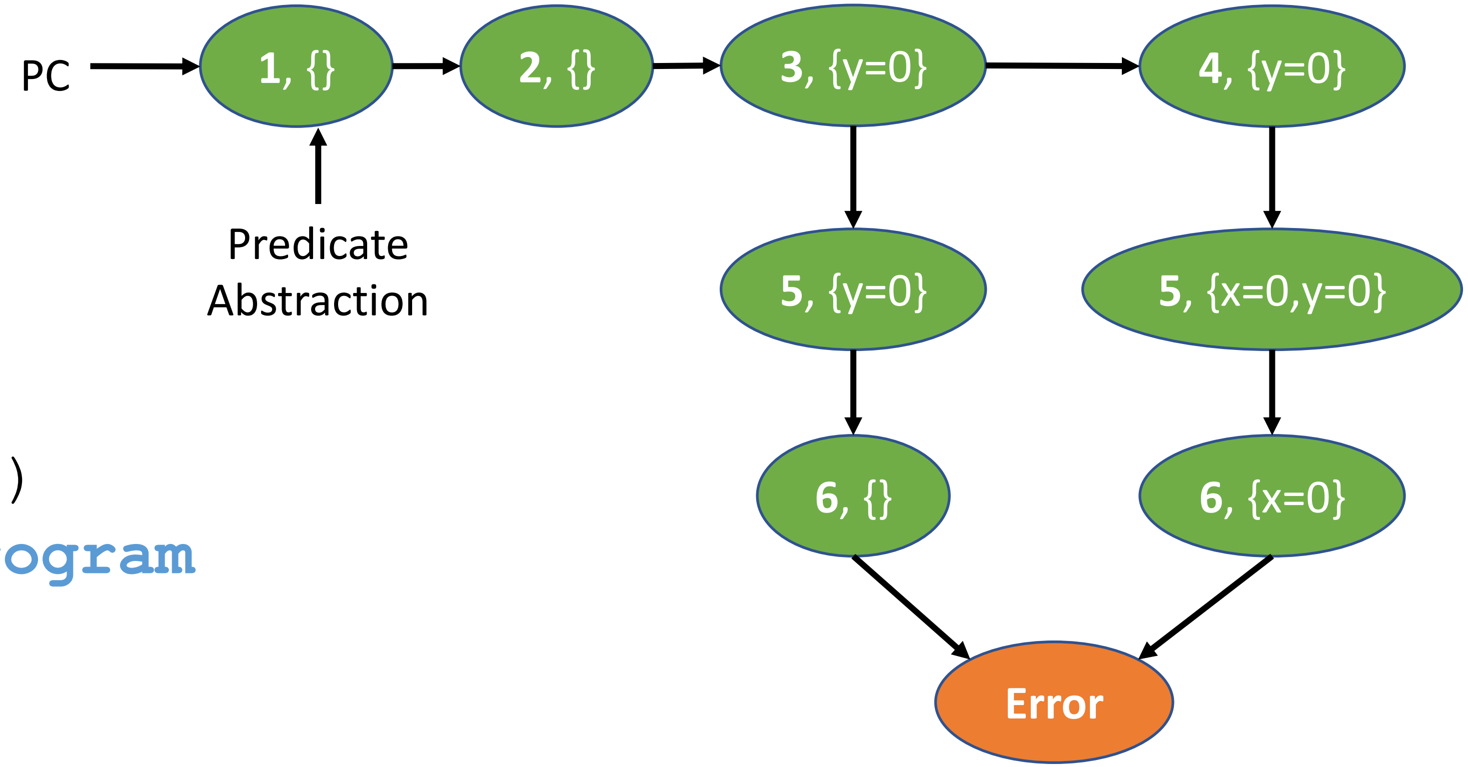
```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

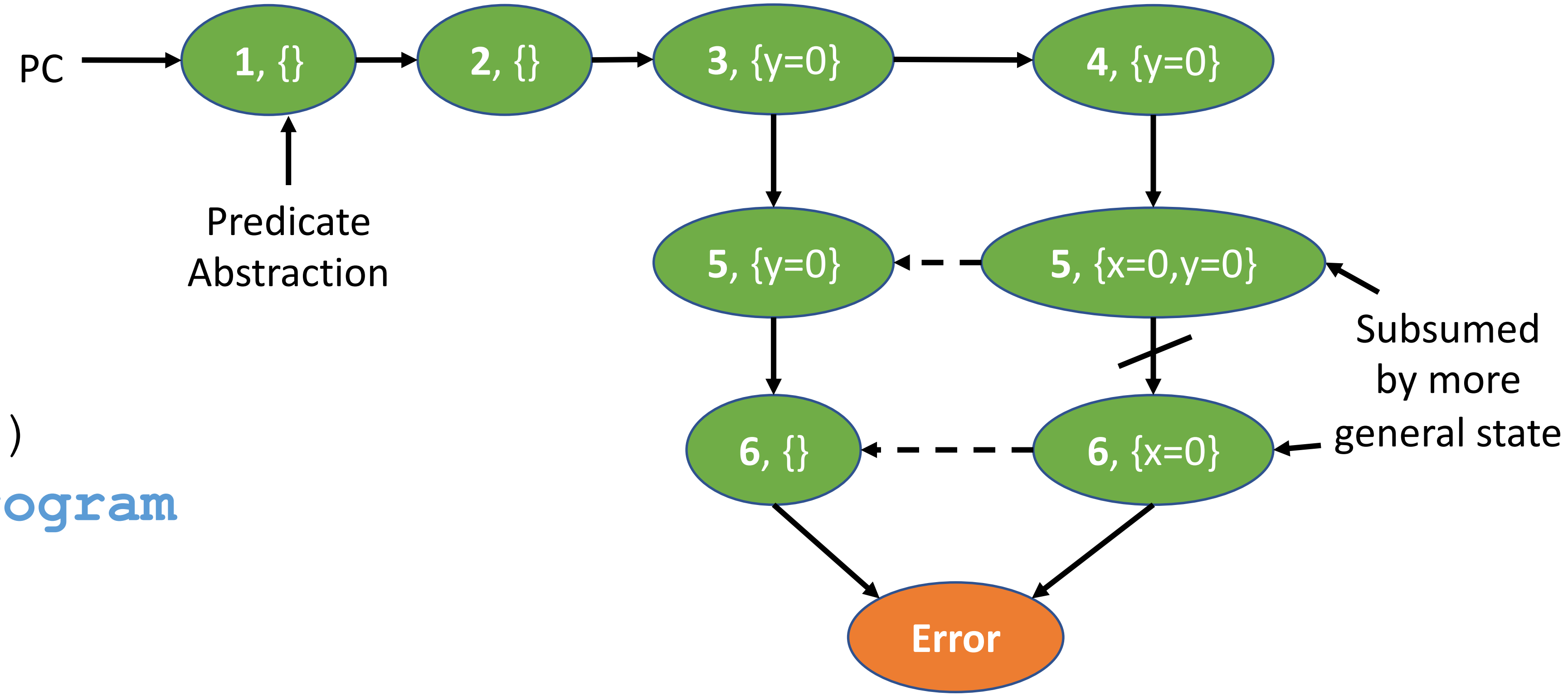
```
main():
  x := *
  y := 0
  if (x < y)
    x := 0
  y := 2
  assert(x != 0)
  // rest of program
```



Cartesian Predicate Abstraction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x = 0, y = 0\}$$

```
   main():
1    x := *
2    y := 0
3    if (x < y)
4      x := 0
5    y := 2
6    assert(x != 0)
7    // rest of program
```



Structure Abstraction

- Generalize predicate abstraction to parameterized programs using *structures*

```
main():
  s := 0
  t := 0
  while (*)
    fork thread

thread():
  local m = t++
  assume (s == m)
  // Critical Section
  s++
```

Structure Abstraction

- Generalize predicate abstraction to parameterized programs using *structures*

```
main():
  s := 0
  t := 0
  while (*)
    fork thread

thread():
  local m = t++
  assume (s == m)
  // Critical Section
  s++
```

Set of Threads → $\mathrm{T} = \{1,2,3\}$

$Q = \{l_i, S_{lt}, M_{lt}\}$ ← Relations over Threads

Structure Abstraction

- Generalize predicate abstraction to parameterized programs using *structures*

```
main():
  s := 0
  t := 0
  while (*)
    fork thread

thread():
  local m = t++
  assume (s == m)
  // Critical Section
  s++
```

Set of Threads → $\mathrm{T} = \{1,2,3\}$

$Q = \{l_i, S_{lt}, M_{lt}\}$ ← Relations over Threads

$l_i(j) \stackrel{\text{def}}{=}$ thread j is at location i

$S_{lt}(j) \stackrel{\text{def}}{=} s < m_j$

$M_{lt}(i,j) \stackrel{\text{def}}{=} m_i < m_j$

Structure Abstraction

- Generalize predicate abstraction to parameterized programs using *structures*

```
  main():
1   s := 0
2   t := 0
3   while (*)
4     fork thread

  thread():
5   local m = t++
6   assume (s == m)
7   // Critical Section
8   s++
```

Set of Threads → $\mathrm{T} = \{1,2,3\}$

$Q = \{l_i, S_{lt}, M_{lt}\}$ ← Relations over Threads

$$l_3(1) \wedge l_6(2) \wedge l_8(3) \wedge S_{lt}(2) \wedge M_{lt}(3,2)$$

$l_i(j) \stackrel{\text{def}}{=}$ thread j is at location i

$$S_{lt}(j) \stackrel{\text{def}}{=} s < m_j$$

$$M_{lt}(i,j) \stackrel{\text{def}}{=} m_i < m_j$$

Sequential vs Parameterized Programs

	Sequential	**Parameterized**
State Space	Sets of predicates	Finite Relational Structures
Subsumption	Subset	Structure Embedding

Overview

1. Use in Multi-threaded Verification

2. Structure Embedding

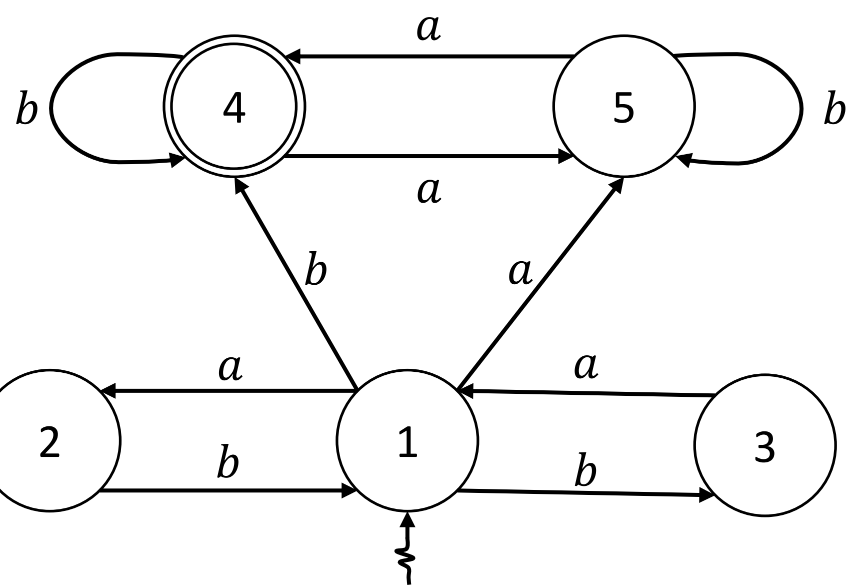
3. MatchEmbeds

Structures

- Finite relational **structure** $\langle \mathcal{U}, \mathcal{R} \rangle$:
  - $\mathcal{U}$: finite universe of elements
  - $\mathcal{R}$: finite set of relations over elements of $\mathcal{U}$

- Examples:
  - State abstractions of multi-threaded programs
  - Graph $\equiv \langle V, edge \rangle$
  - NFA $\equiv \langle S, \{final, start\} \cup \{\Delta_a : a \in \Sigma\} \rangle$

Structures



$$\mathfrak{F} \stackrel{\text{def}}{=} \langle \{1,2,3,4,5\}, Start, Final, \Delta_a, \Delta_b \rangle$$

where:

$$Start \stackrel{\text{def}}{=} \{1\}$$

$$Final \stackrel{\text{def}}{=} \{4\}$$

$$\Delta_a \stackrel{\text{def}}{=} \{\langle 1,2\rangle, \langle 1,5\rangle, \langle 3,1\rangle, \langle 4.5\rangle, \langle 5,4\rangle\}$$

$$\Delta_b \stackrel{\text{def}}{=} \{\langle 1,3\rangle, \langle 1,4\rangle, \langle 2,1\rangle, \langle 4,4\rangle, \langle 5,5\rangle\}$$

Structure Embedding

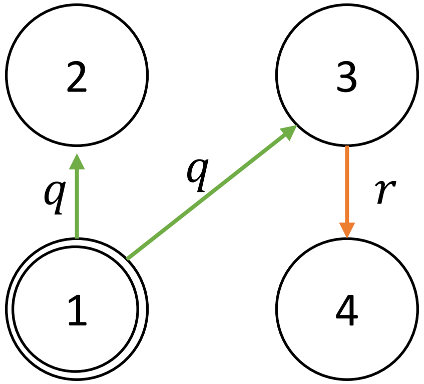
$$\mathfrak{A} \stackrel{\text{def}}{=} \langle\{1,2,3,4\}, p^{\mathfrak{A}}, q^{\mathfrak{A}}, r^{\mathfrak{A}}\rangle$$

$$p^{\mathfrak{A}} \stackrel{\text{def}}{=} \{1\}$$

$$q^{\mathfrak{A}} \stackrel{\text{def}}{=} \{\langle 1,2\rangle, \langle 1,3\rangle\}$$

$$r^{\mathfrak{A}} \stackrel{\text{def}}{=} \{\langle 3,4\rangle\}$$

$\mathfrak{A}$:



Structure Embedding

$\mathfrak{A} \overset{\text{def}}{=} \langle \{1,2,3,4\}, p^{\mathfrak{A}}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \rangle$

$p^{\mathfrak{A}} \overset{\text{def}}{=} \{1\}$

$q^{\mathfrak{A}} \overset{\text{def}}{=} \{\langle 1,2 \rangle, \langle 1,3 \rangle\}$

$r^{\mathfrak{A}} \overset{\text{def}}{=} \{\langle 3,4 \rangle\}$

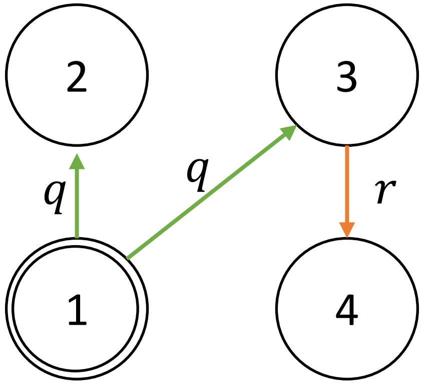
$\mathfrak{B} \overset{\text{def}}{=} \langle \{1,2,3,4\}, p^{\mathfrak{B}}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \rangle$

$p^{\mathfrak{B}} \overset{\text{def}}{=} \{1,3\}$

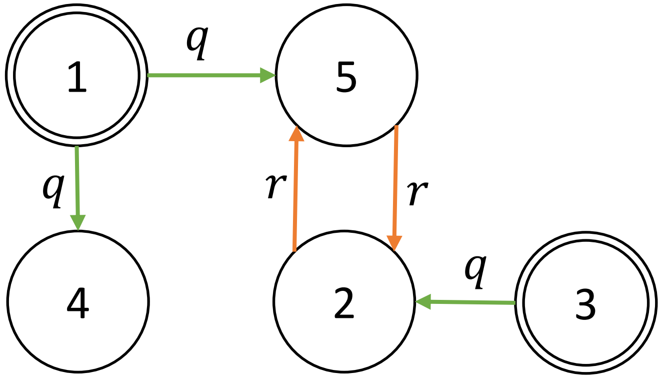
$q^{\mathfrak{B}} \overset{\text{def}}{=} \{\langle 1,4 \rangle, \langle 1,5 \rangle, \langle 3,2 \rangle\}$

$r^{\mathfrak{B}} \overset{\text{def}}{=} \{\langle 2,5 \rangle, \langle 5,2 \rangle\}$

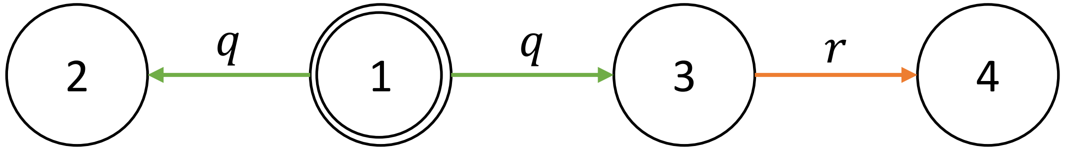
$\mathfrak{A}$:



$\mathfrak{B}$:



Structure Embedding

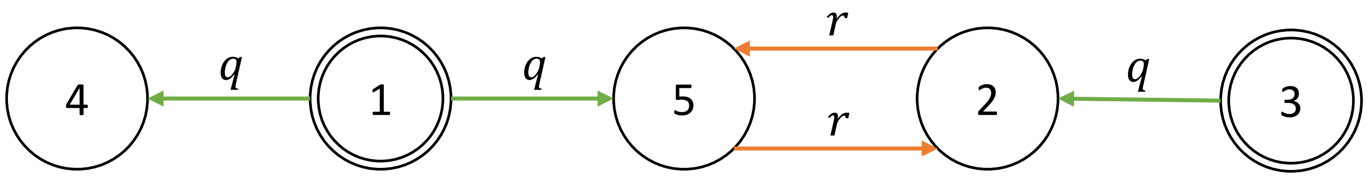


$$\mathfrak{A} \stackrel{\text{def}}{=} \left\langle \{1,2,3,4\}, p^{\mathfrak{A}}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \right\rangle$$

$$p^{\mathfrak{A}} \stackrel{\text{def}}{=} \{1\}$$

$$q^{\mathfrak{A}} \stackrel{\text{def}}{=} \{\langle 1,2\rangle, \langle 1,3\rangle\}$$

$$r^{\mathfrak{A}} \stackrel{\text{def}}{=} \{\langle 3,4\rangle\}$$



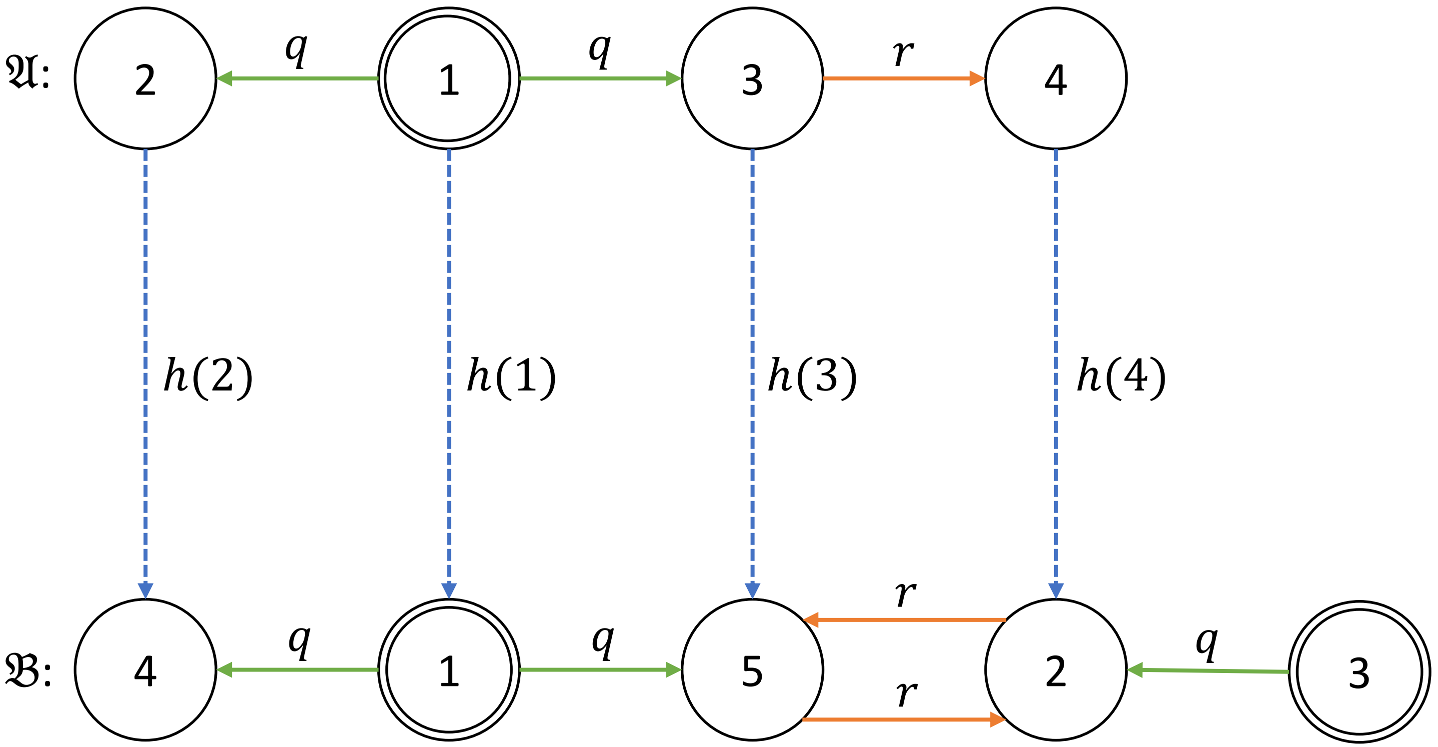
$$\mathfrak{B} \stackrel{\text{def}}{=} \left\langle \{1,2,3,4\}, p^{\mathfrak{B}}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \right\rangle$$

$$p^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,3\}$$

$$q^{\mathfrak{B}} \stackrel{\text{def}}{=} \{\langle 1,4\rangle, \langle 1,5\rangle, \langle 3,2\rangle\}$$

$$r^{\mathfrak{B}} \stackrel{\text{def}}{=} \{\langle 2,5\rangle, \langle 5,2\rangle\}$$

Structure Embedding



$\mathfrak{A} \stackrel{\text{def}}{\equiv} \left\langle \{1,2,3,4\}, p^{\mathfrak{A}}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \right\rangle$

$p^{\mathfrak{A}} \stackrel{\text{def}}{\equiv} \{1\}$

$q^{\mathfrak{A}} \stackrel{\text{def}}{\equiv} \{\langle 1,2 \rangle, \langle 1,3 \rangle\}$

$r^{\mathfrak{A}} \stackrel{\text{def}}{\equiv} \{\langle 3,4 \rangle\}$

$\mathfrak{B} \stackrel{\text{def}}{\equiv} \left\langle \{1,2,3,4\}, p^{\mathfrak{B}}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \right\rangle$

$p^{\mathfrak{B}} \stackrel{\text{def}}{\equiv} \{1,3\}$

$q^{\mathfrak{B}} \stackrel{\text{def}}{\equiv} \{\langle 1,4 \rangle, \langle 1,5 \rangle, \langle 3,2 \rangle\}$

$r^{\mathfrak{B}} \stackrel{\text{def}}{\equiv} \{\langle 2,5 \rangle, \langle 5,2 \rangle\}$

Structure Embedding Problem

- Given two structures $\mathfrak{A}$ and $\mathfrak{B}$, is there an injective homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$?
- Is NP-complete
  - Generalizes subgraph isomorphism
- Verifying a program using structure abstraction
  - May take thousands of embedding instances
  - Most instances are small
  - Many instances are monadic

Overview

1. Use in Multi-threaded Verification

2. Structure Embedding

3. MatchEmbeds

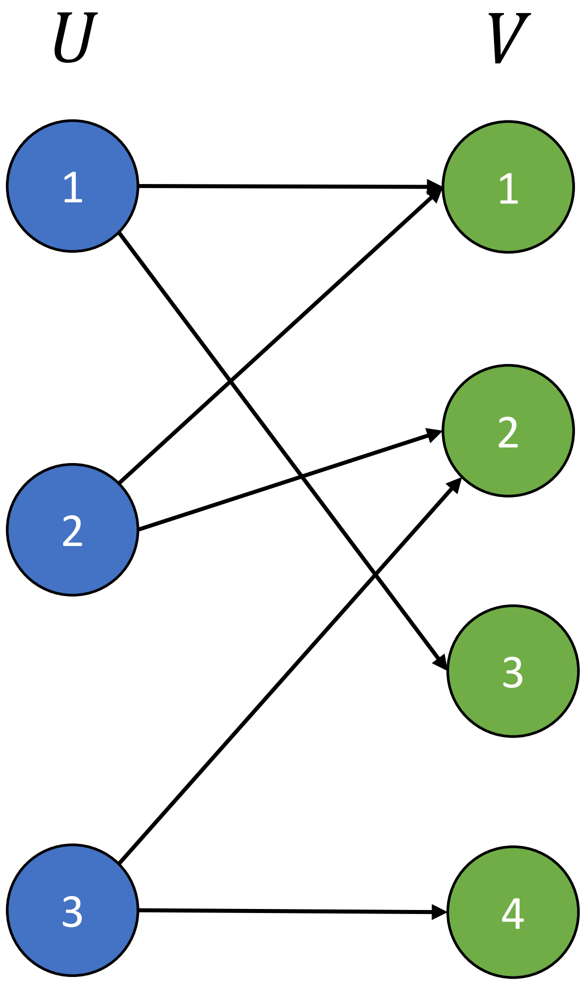
MatchEmbeds

MatchEmbeds

- Solves the structure embedding problem
  - Polytime for monadic case reduction to bipartite graph matching
  - Quick for instances from verification
- Backtracking Search
  - Construct bipartite graph
  - Globally searches over space of total matchings
  - Locally chooses edges in a matching to direct search

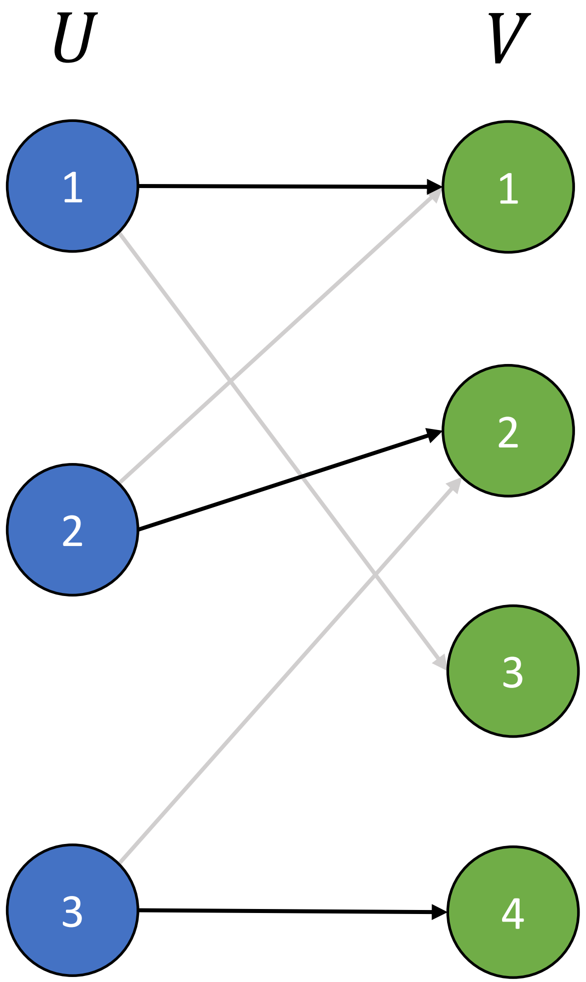
Bipartite Graphs

- Bipartite Graphs, $G = \langle U, V, E \rangle$
  - U and V are disjoint
  - $E \subseteq U \times V$



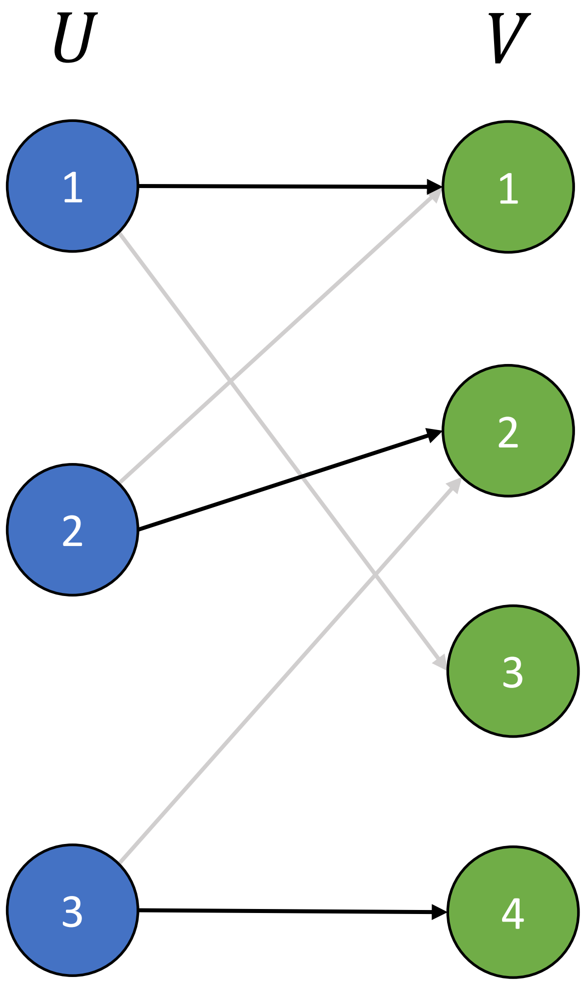
Bipartite Graphs

- Bipartite Graphs, $G = \langle U, V, E \rangle$
  - U and V are disjoint
  - $E \subseteq U \times V$
- Matching, $M \subseteq E$
  - Each vertex is incident to at most one edge in M
  - Maximum matching – highest cardinality
  - Total matching – all vertices in U incident to M



Bipartite Graphs

- Bipartite Graphs, $G = \langle U, V, E \rangle$
  - U and V are disjoint
  - $E \subseteq U \times V$
- Matching, $M \subseteq E$
  - Each vertex is incident to at most one edge in M
  - Maximum matching – highest cardinality
  - Total matching – all vertices in U incident to M
- Observe: total matchings correspond to injective functions $U \to V$

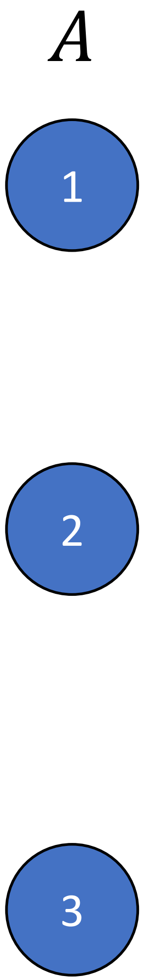


Monadic Case

$$\mathfrak{A} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \rangle$$

$$q^{\mathfrak{A}} \stackrel{\text{def}}{=} \{1\}$$

$$r^{\mathfrak{A}} \stackrel{\text{def}}{=} \{2,3\}$$



Monadic Case

$$\mathfrak{A} \overset{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \rangle$$

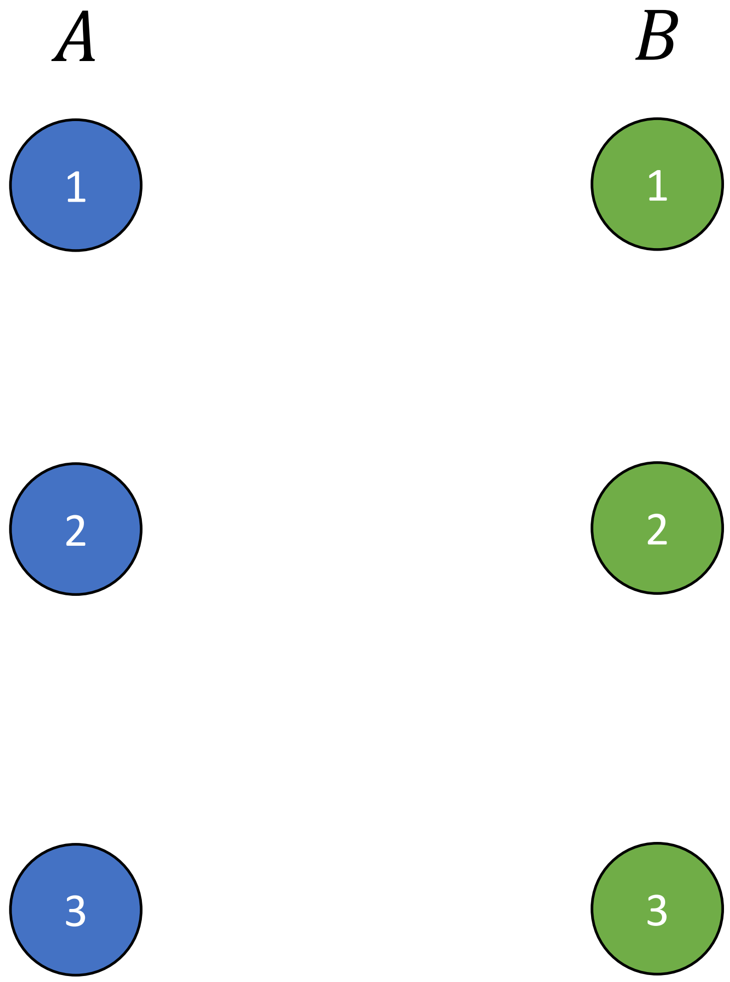
$$q^{\mathfrak{A}} \overset{\text{def}}{=} \{1\}$$

$$r^{\mathfrak{A}} \overset{\text{def}}{=} \{2,3\}$$

$$\mathfrak{B} \overset{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \rangle$$

$$q^{\mathfrak{B}} \overset{\text{def}}{=} \{1,2,3\}$$

$$r^{\mathfrak{B}} \overset{\text{def}}{=} \{1,3\}$$



Monadic Case

$$\mathfrak{A} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \rangle$$

$q^{\mathfrak{A}} \stackrel{\text{def}}{=} \{1\}$

$r^{\mathfrak{A}} \stackrel{\text{def}}{=} \{2,3\}$

$sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q\}$

$sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{r\}$

$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{r\}$

$$\mathfrak{B} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \rangle$$

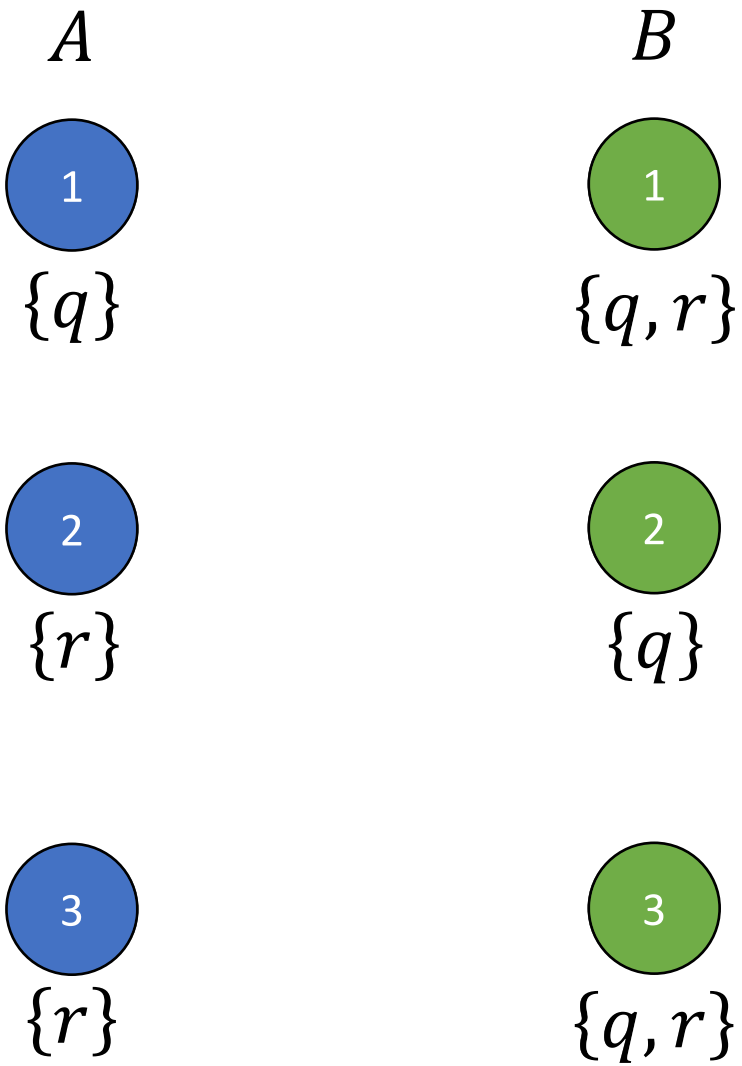
$q^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,2,3\}$

$r^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,3\}$

$sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q, r\}$

$sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{q\}$

$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{q, r\}$



Monadic Case

$\mathfrak{A} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \rangle$

$q^{\mathfrak{A}} \stackrel{\text{def}}{=} \{1\}$ $\quad sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q\}$

$r^{\mathfrak{A}} \stackrel{\text{def}}{=} \{2,3\}$ $\quad sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{r\}$

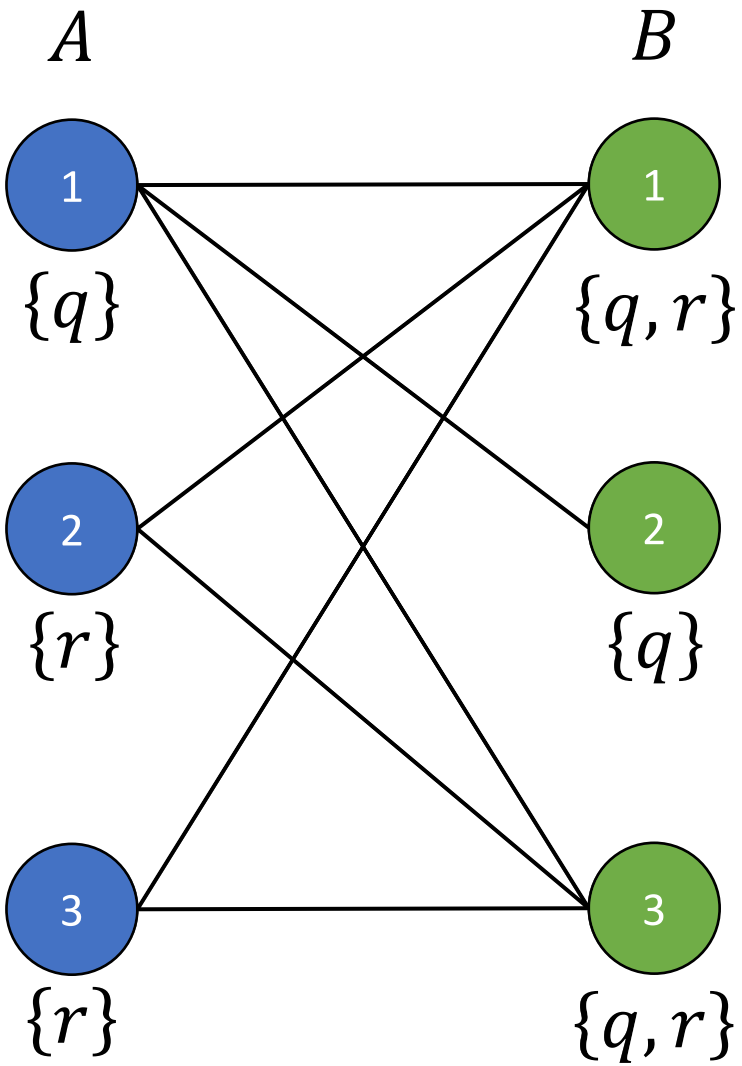
$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{r\}$

$\mathfrak{B} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \rangle$

$q^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,2,3\}$ $\quad sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q, r\}$

$r^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,3\}$ $\quad sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{q\}$

$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{q, r\}$



Monadic Case

$\mathfrak{A} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \rangle$

$q^{\mathfrak{A}} \stackrel{\text{def}}{=} \{1\}$

$r^{\mathfrak{A}} \stackrel{\text{def}}{=} \{2,3\}$

$sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q\}$

$sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{r\}$

$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{r\}$

$\mathfrak{B} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \rangle$

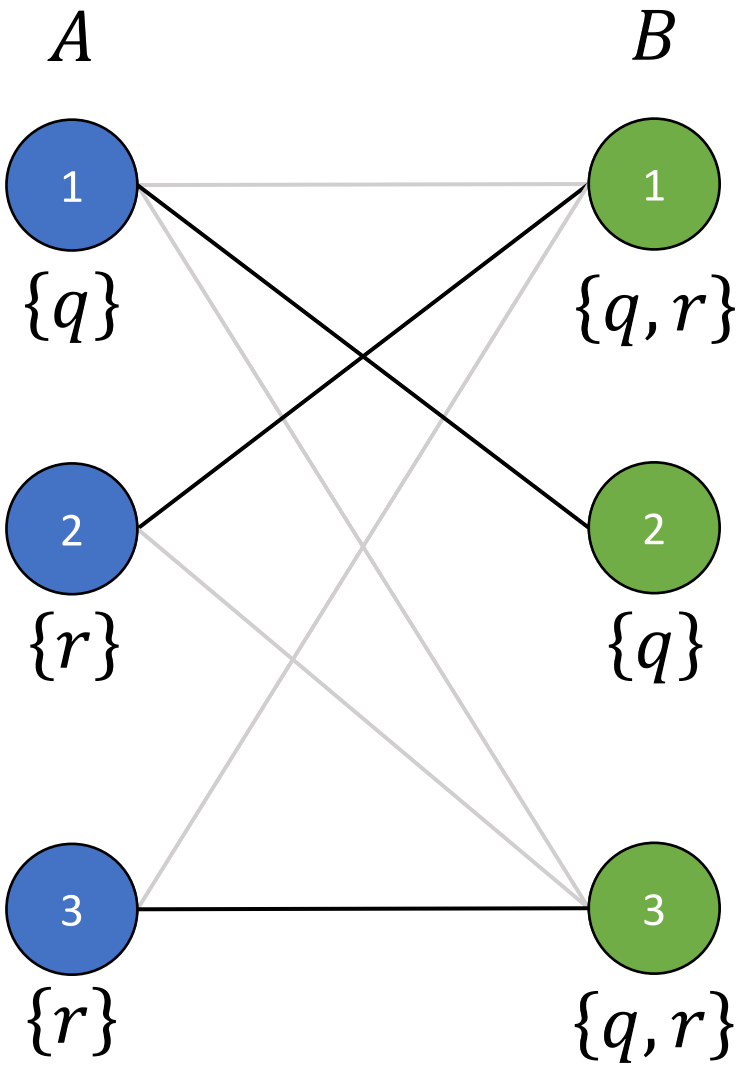
$q^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,2,3\}$

$r^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,3\}$

$sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q, r\}$

$sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{q\}$

$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{q, r\}$



Maximum Matchings

$$M_1 \stackrel{\text{def}}{=} \left\{ \begin{matrix} \langle 1,2 \rangle, \\ \langle 2,1 \rangle, \\ \langle 3,3 \rangle \end{matrix} \right\}$$

Monadic Case

$\mathfrak{A} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{A}}, r^{\mathfrak{A}} \rangle$

$q^{\mathfrak{A}} \stackrel{\text{def}}{=} \{1\}$ $\quad sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q\}$

$r^{\mathfrak{A}} \stackrel{\text{def}}{=} \{2,3\}$ $\quad sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{r\}$

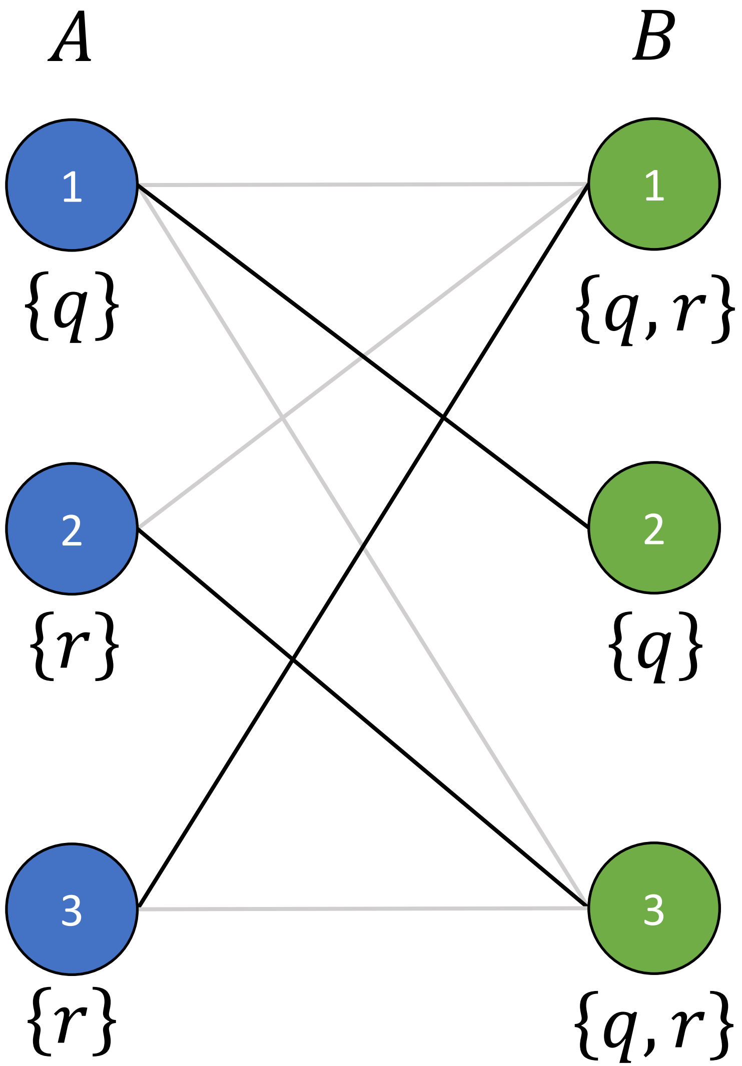
$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{r\}$

$\mathfrak{B} \stackrel{\text{def}}{=} \langle \{1,2,3\}, q^{\mathfrak{B}}, r^{\mathfrak{B}} \rangle$

$q^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,2,3\}$ $\quad sig(\mathfrak{A}, 1) \stackrel{\text{def}}{=} \{q, r\}$

$r^{\mathfrak{B}} \stackrel{\text{def}}{=} \{1,3\}$ $\quad sig(\mathfrak{A}, 2) \stackrel{\text{def}}{=} \{q\}$

$sig(\mathfrak{A}, 3) \stackrel{\text{def}}{=} \{q, r\}$



Maximum Matchings

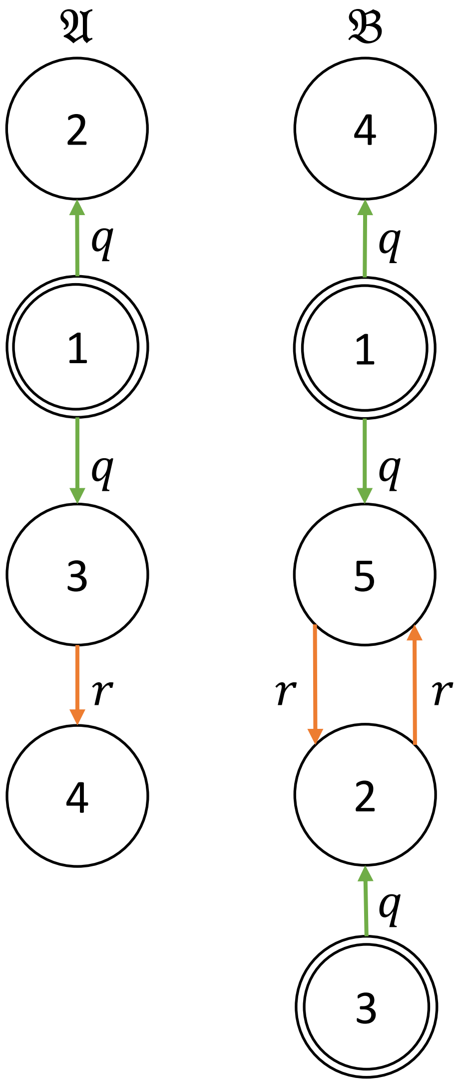
$$M_1 \stackrel{\text{def}}{=} \left\{ \begin{matrix} \langle 1,2 \rangle, \\ \langle 2,1 \rangle, \\ \langle 3,3 \rangle \end{matrix} \right\}$$

$$M_2 \stackrel{\text{def}}{=} \left\{ \begin{matrix} \langle 1,2 \rangle, \\ \langle 2,3 \rangle, \\ \langle 3,1 \rangle \end{matrix} \right\}$$

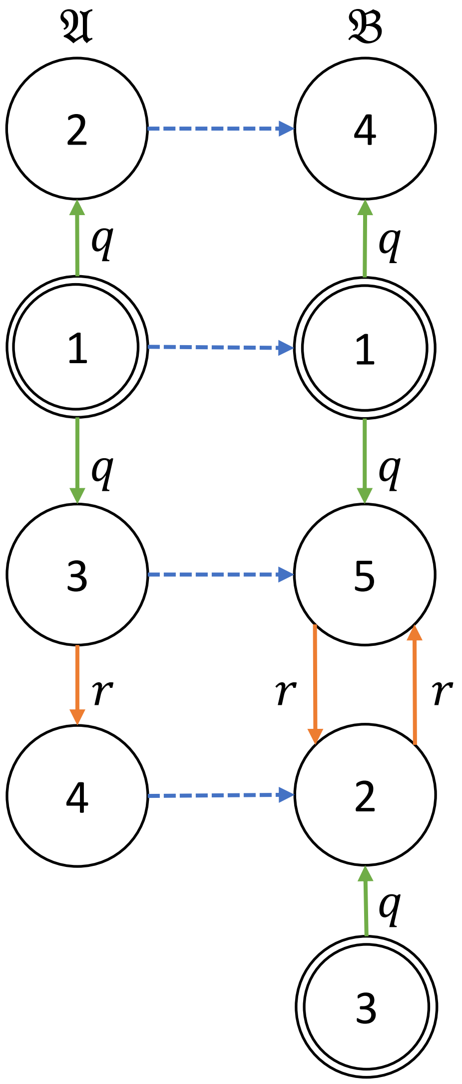
Monadic Case

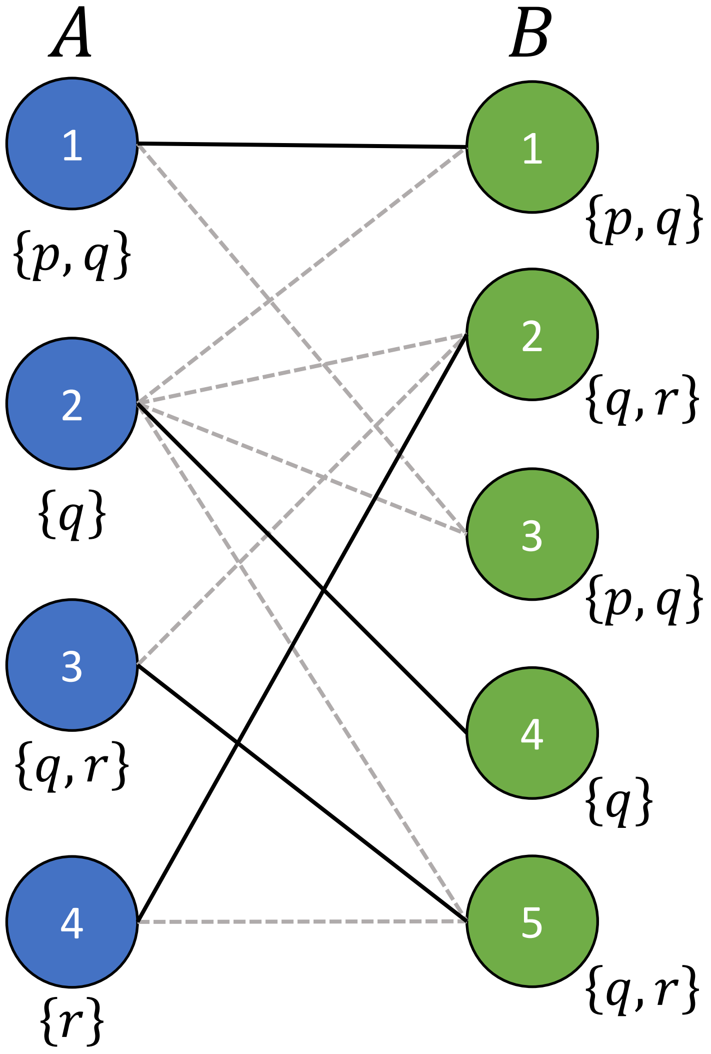
- Structures, $\mathfrak{A}$ and $\mathfrak{B}$, where each relation has arity 1
- **Signature Graph** $(Sig(\mathfrak{A}, \mathfrak{B}))$
  - Draws edges from a to b if a may map to b
  - Total matchings on $Sig(\mathfrak{A}, \mathfrak{B})$ are embeddings
- Structure embedding takes $O(|A||B|\sqrt{|A| + |B|})$ [Hopcroft and Karp. 1973]

General Case

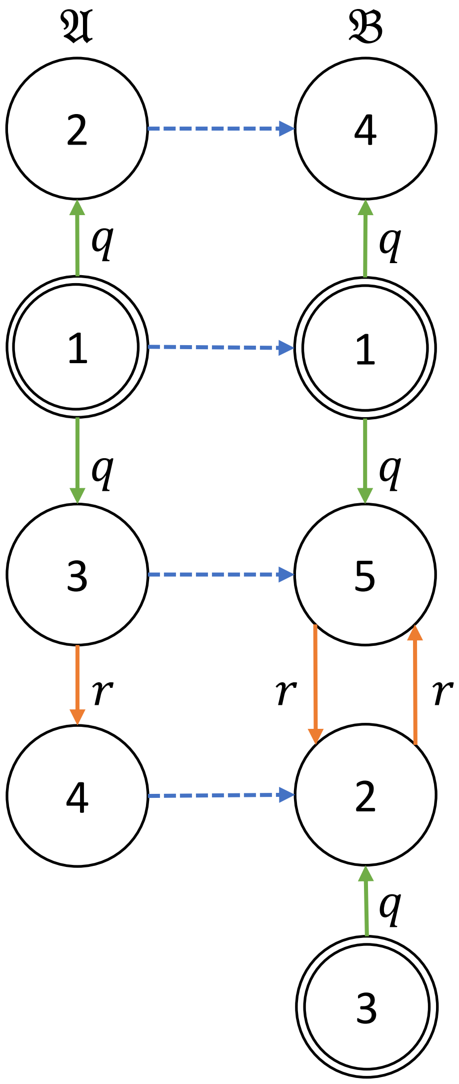


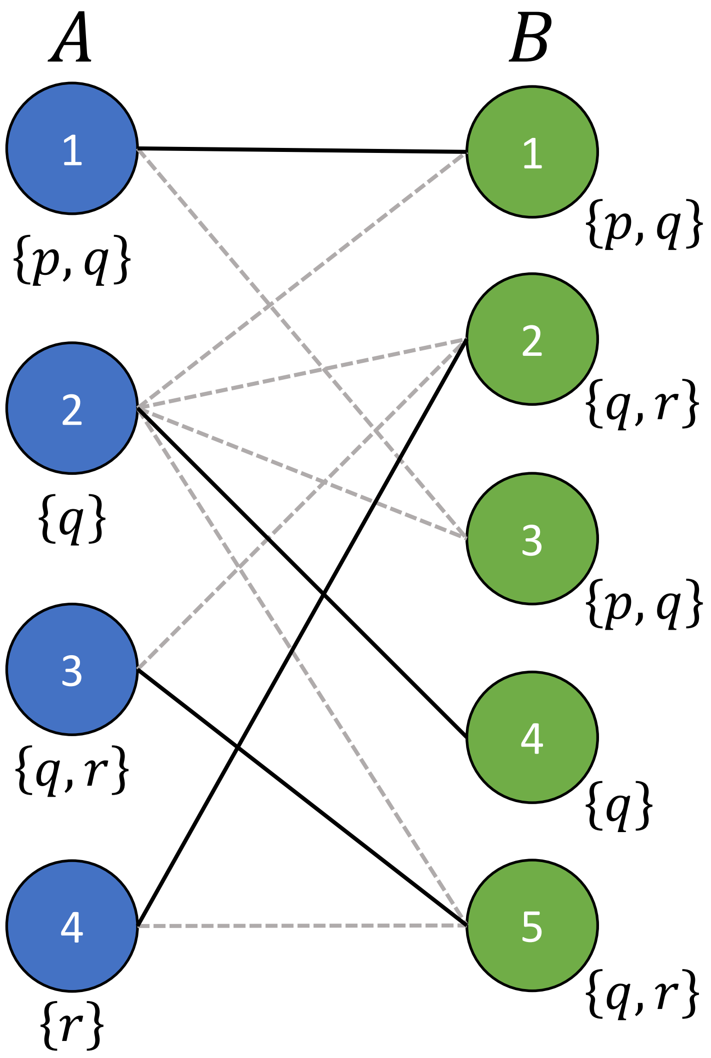
General Case





General Case





$M_1 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

$\boldsymbol{M_2} \stackrel{\text{def}}{=} \{\langle \mathbf{1}, \mathbf{1}\rangle, \langle \mathbf{2}, \mathbf{4}\rangle, \langle \mathbf{3}, \mathbf{5}\rangle, \langle \mathbf{4}, \mathbf{2}\rangle\}$

$M_3 \stackrel{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

$M_4 \stackrel{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,4\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

$M_5 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

$M_6 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,3\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

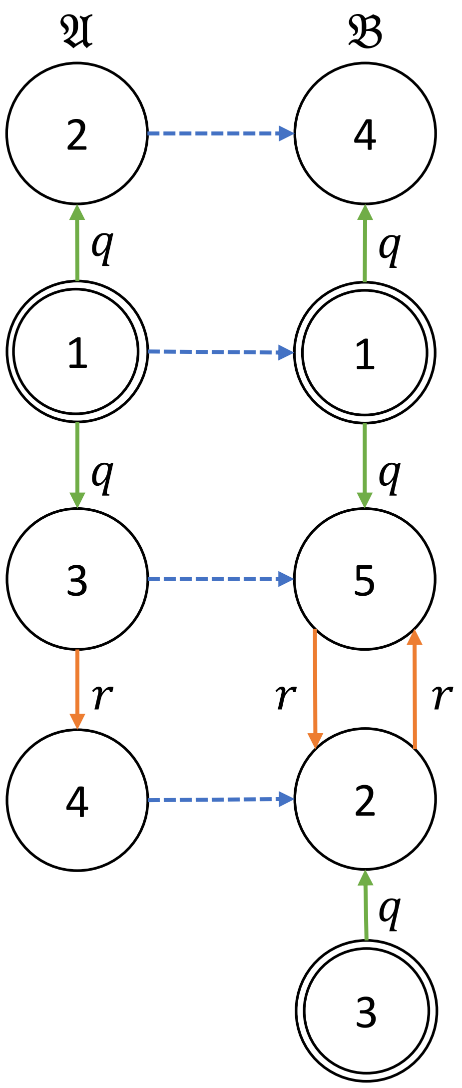
$M_7 \stackrel{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,1\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

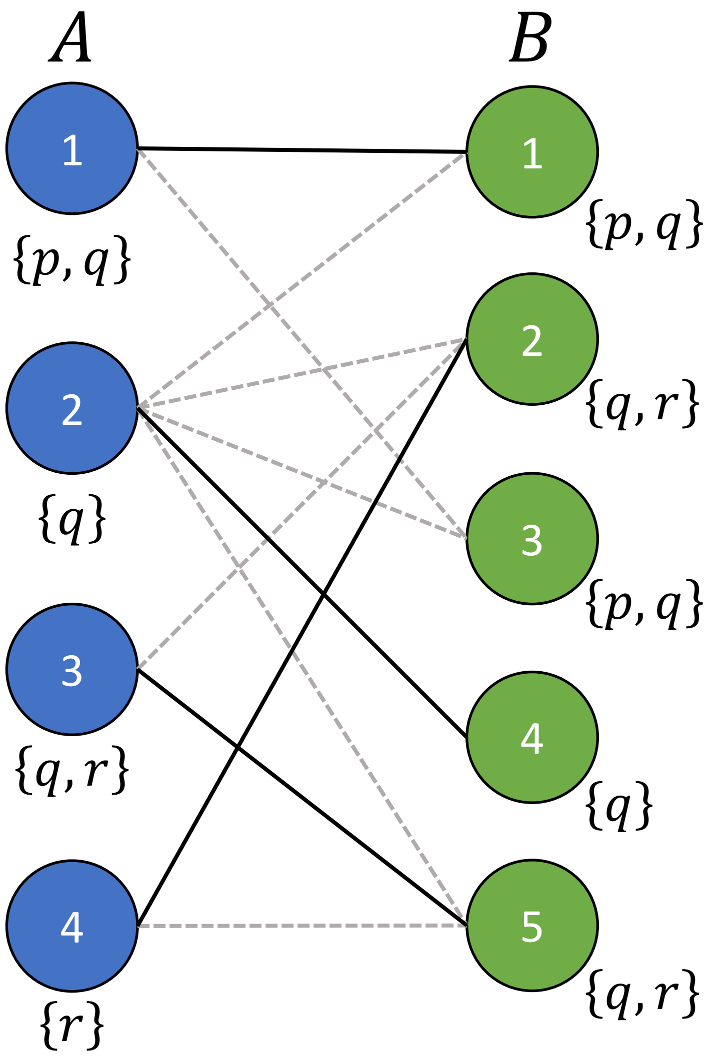
$M_8 \stackrel{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,1\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

MatchEmbeds

- Inspired by monadic reduction to bipartite graph matching
  - If M is a structure embedding then $M \subseteq E$ is a total matching of $Sig(\mathfrak{A}, \mathfrak{B})$
  - Ensures monadic case remains polytime
- Backtracking search algorithm over total matchings
  1. Remove *inconsistent* edges from graph
  2. Compute maximum matching
  3. Check for *conflicts*
  4. Decide on edges in matching and recurse

General Case





$M_1 \overset{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

$\boldsymbol{M_2 \overset{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}}$

$M_3 \overset{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

$M_4 \overset{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,4\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

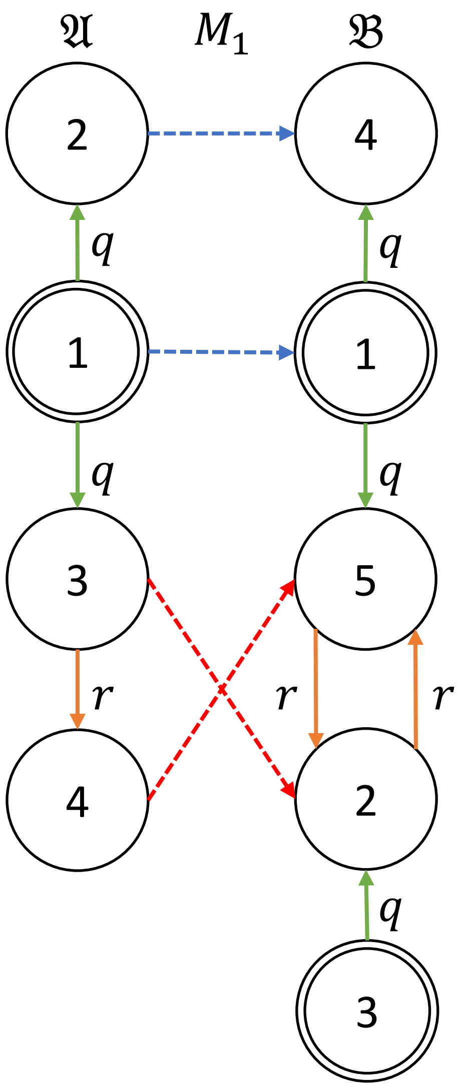
$M_5 \overset{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

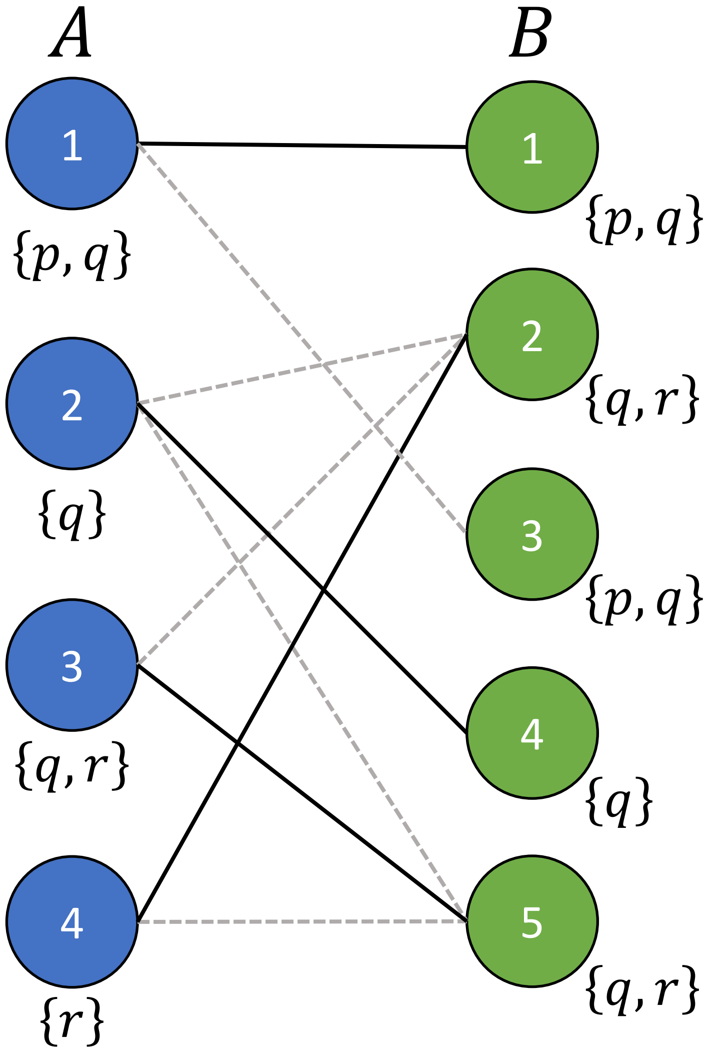
$M_6 \overset{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,3\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

$M_7 \overset{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,1\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

$M_8 \overset{\text{def}}{=} \{\langle 1,3\rangle, \langle 2,1\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

MatchEmbeds

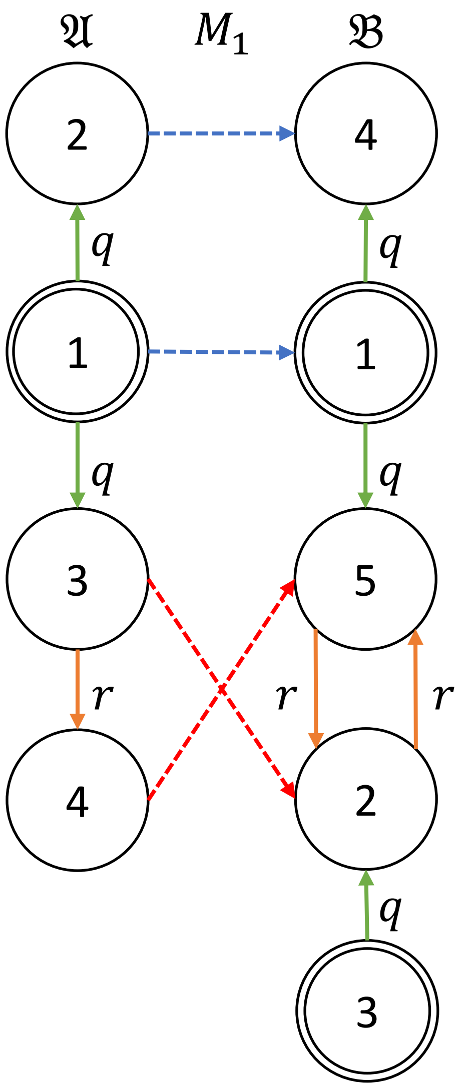


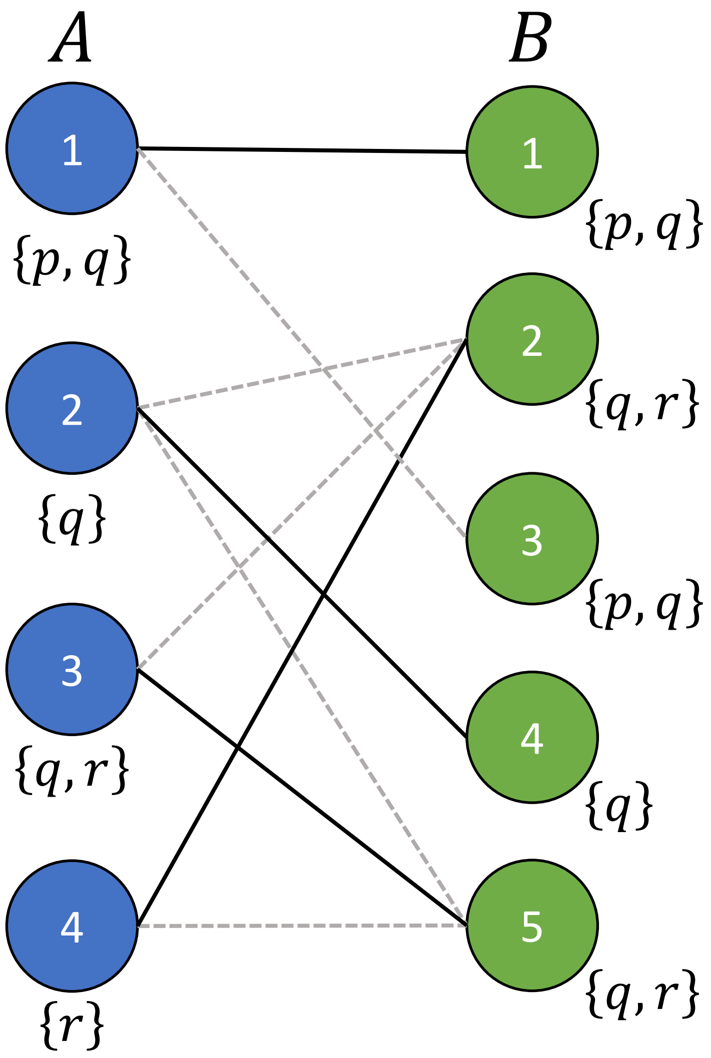


Compute Matching

$M_1 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

MatchEmbeds

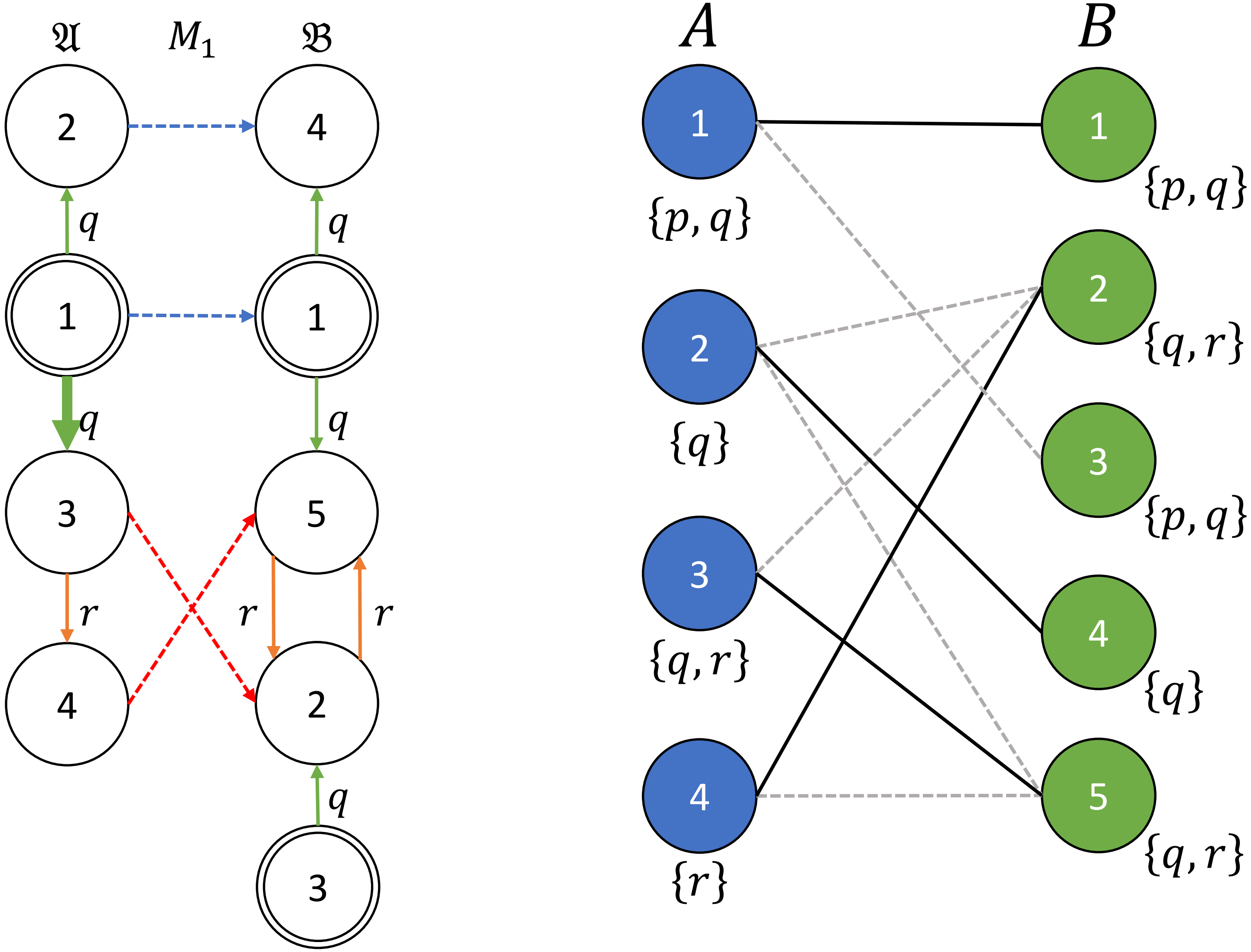




Compute Conflict Set

$M_1 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

MatchEmbeds



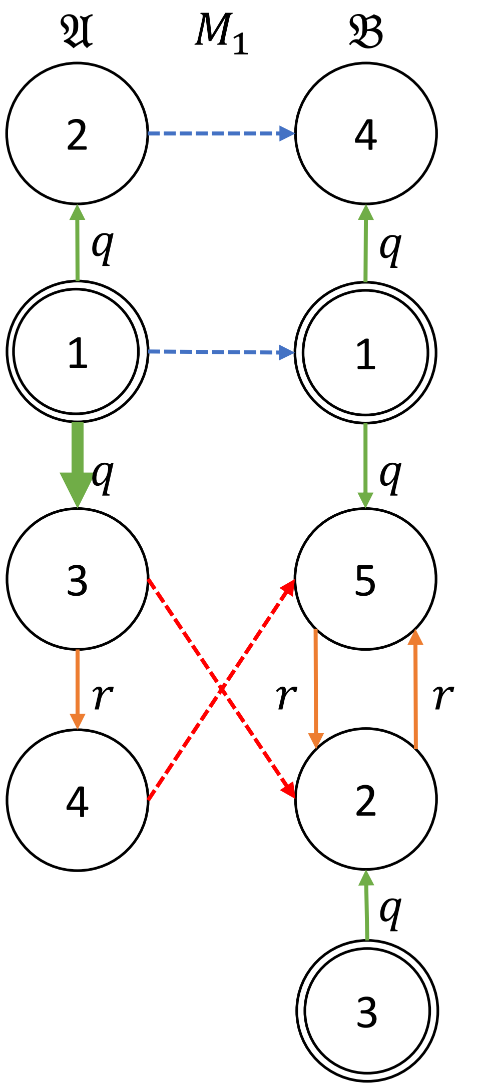
Compute Conflict Set

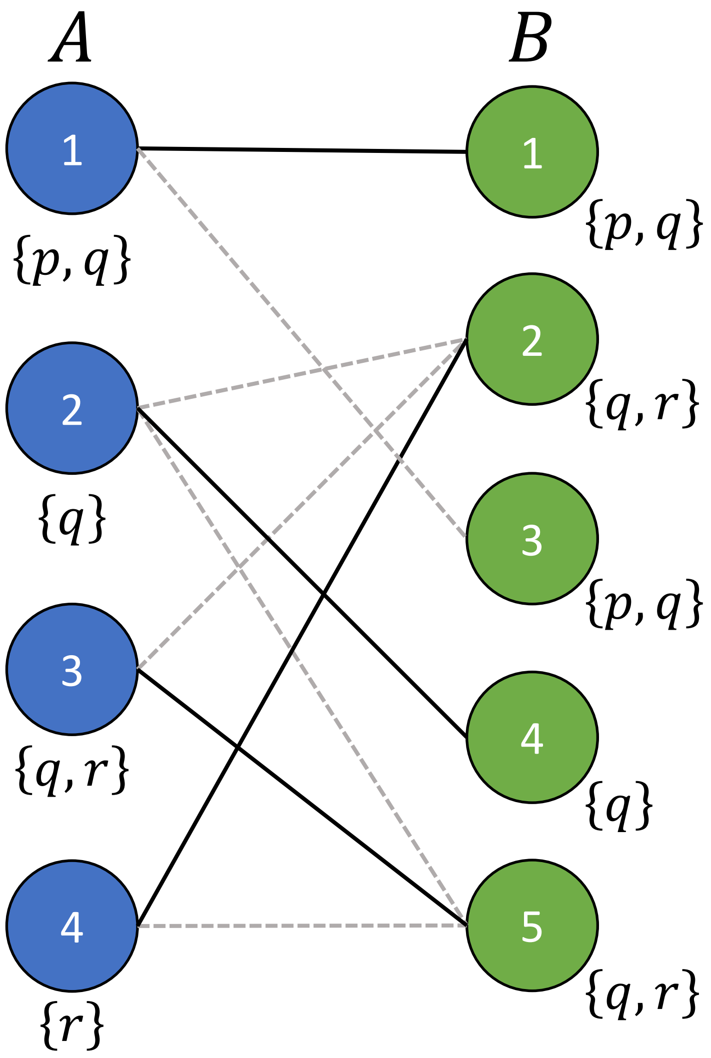
$M_1 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

$Conflict(M_1) \stackrel{\text{def}}{=} \{q(1,3)\}$

Set of predicates in 𝔄 not preserved by M

MatchEmbeds





Compute Decisions

$M_1 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,2\rangle, \langle 4,5\rangle\}$

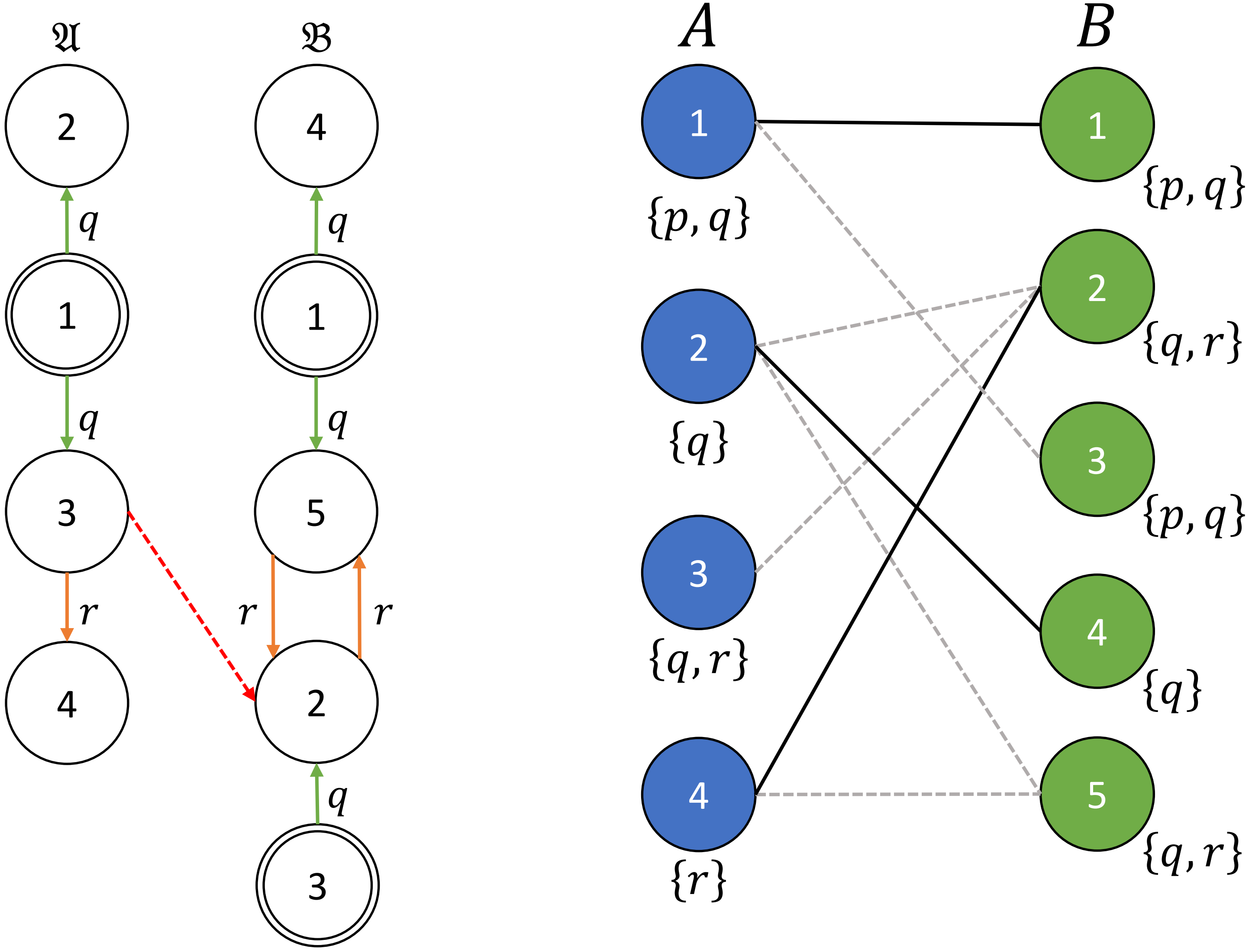
$Conflict(M_1) \stackrel{\text{def}}{=} \{q(1,3)\}$

$Decisions(M_1) \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 3,2\rangle\}$

Edges $\langle a, b\rangle \in M_1$ s.t.

1. a is in a conflict
2. $degree(a) > 1$

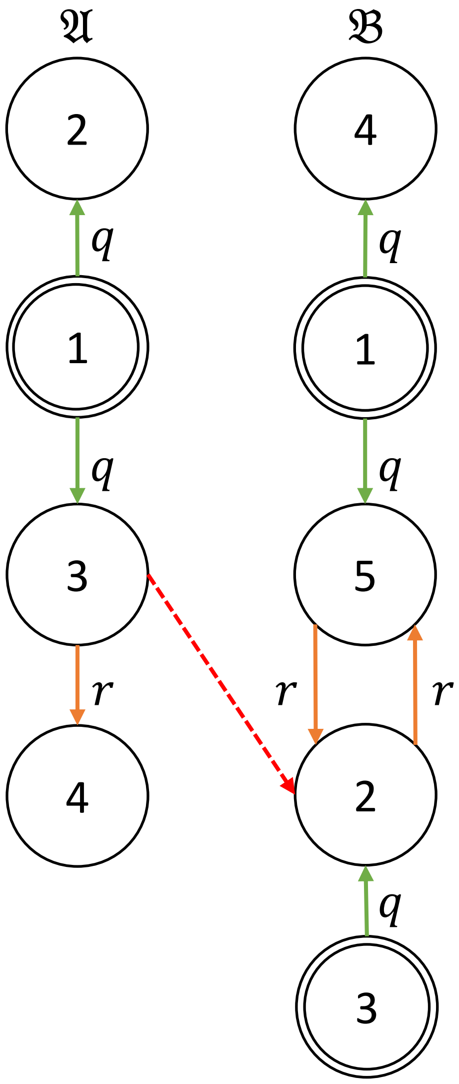
MatchEmbeds

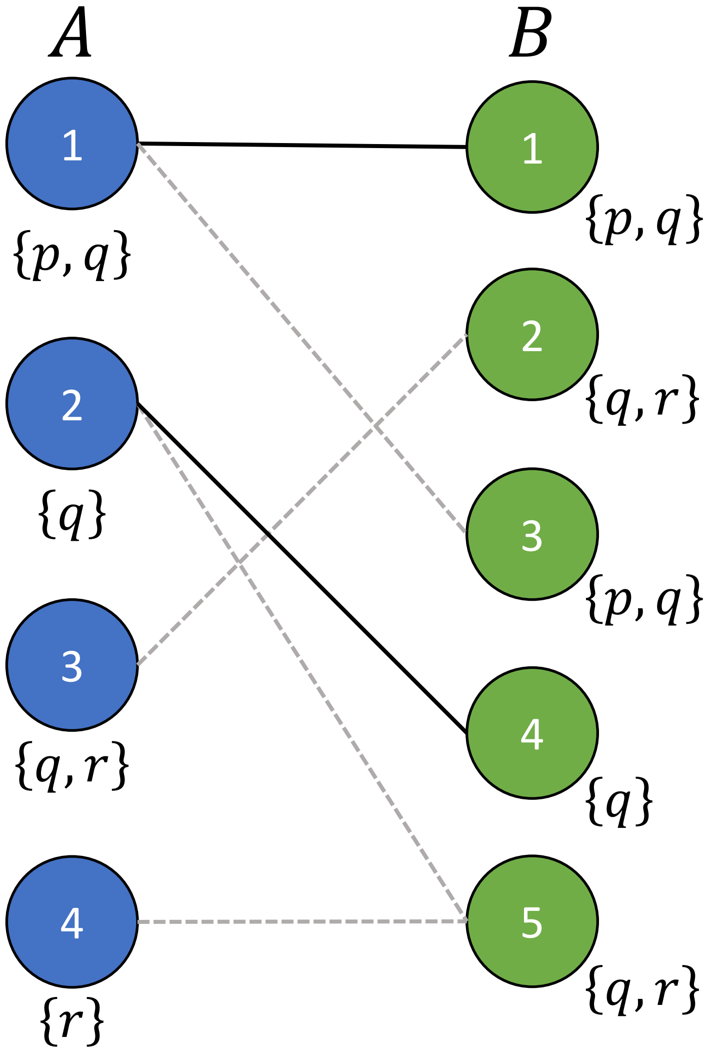


Decide $[3 \longmapsto 2]$

- Remove $\langle 3,5 \rangle$

MatchEmbeds

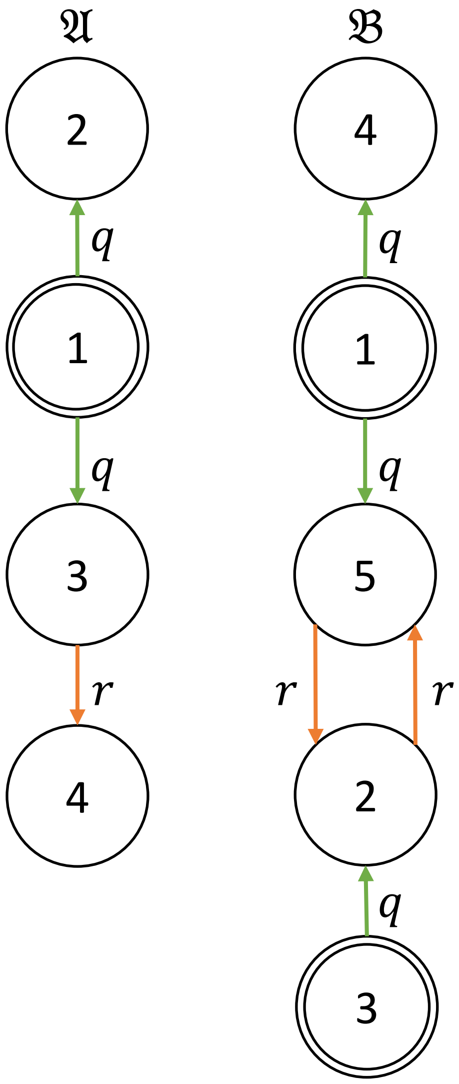


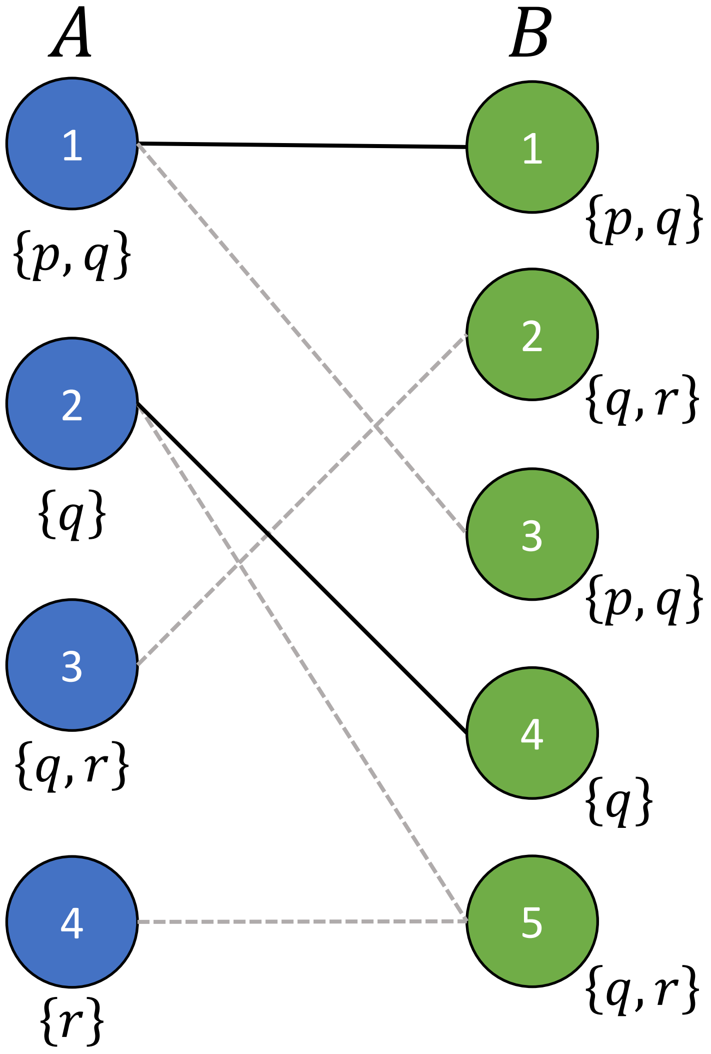


Decide $[3 \longmapsto 2]$

- Remove $\langle 3,5 \rangle, \langle 2,2 \rangle, \langle 4,2 \rangle$
- Compute consistent sub-graph

Maximum Consistent Sub-Graph



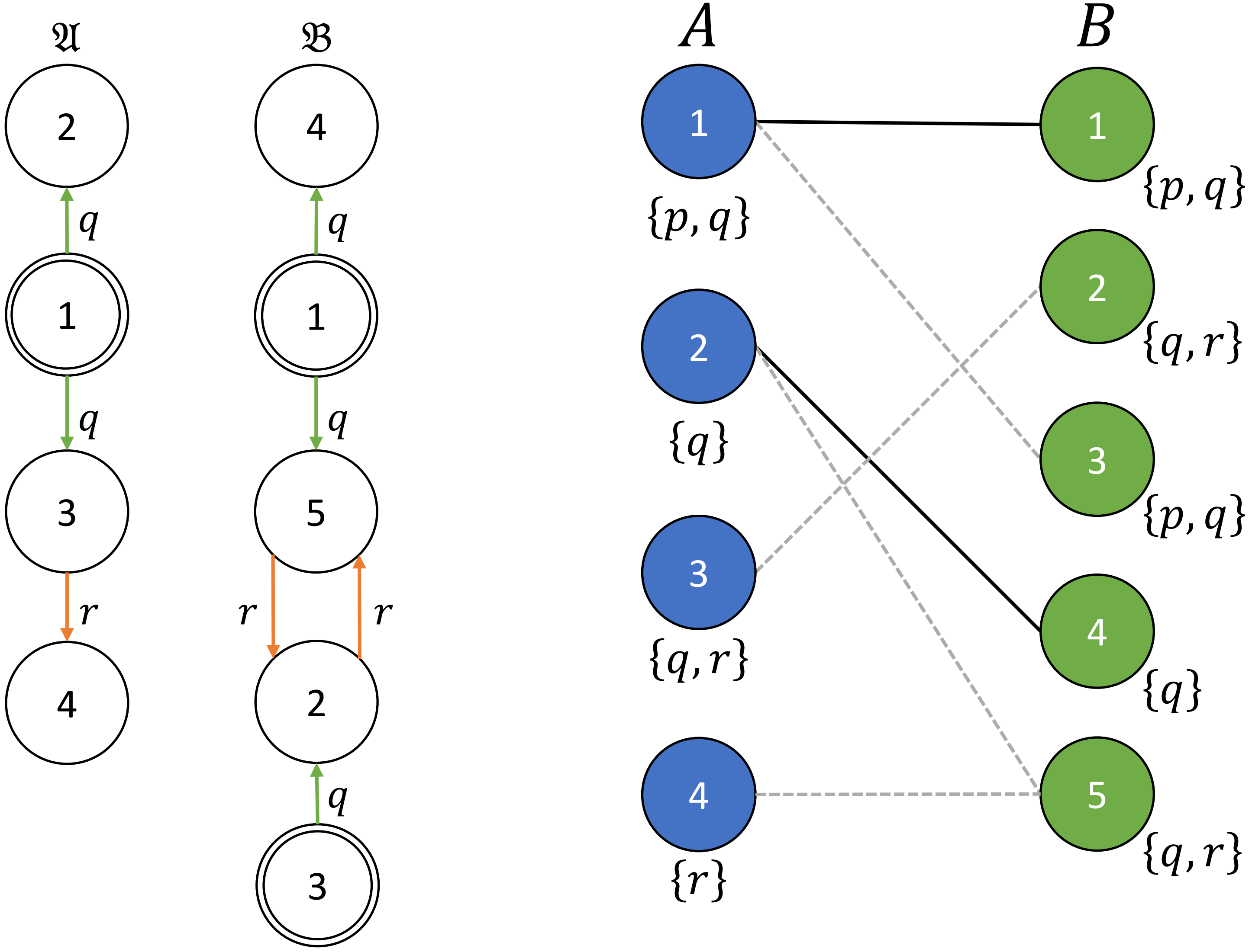


Goals:

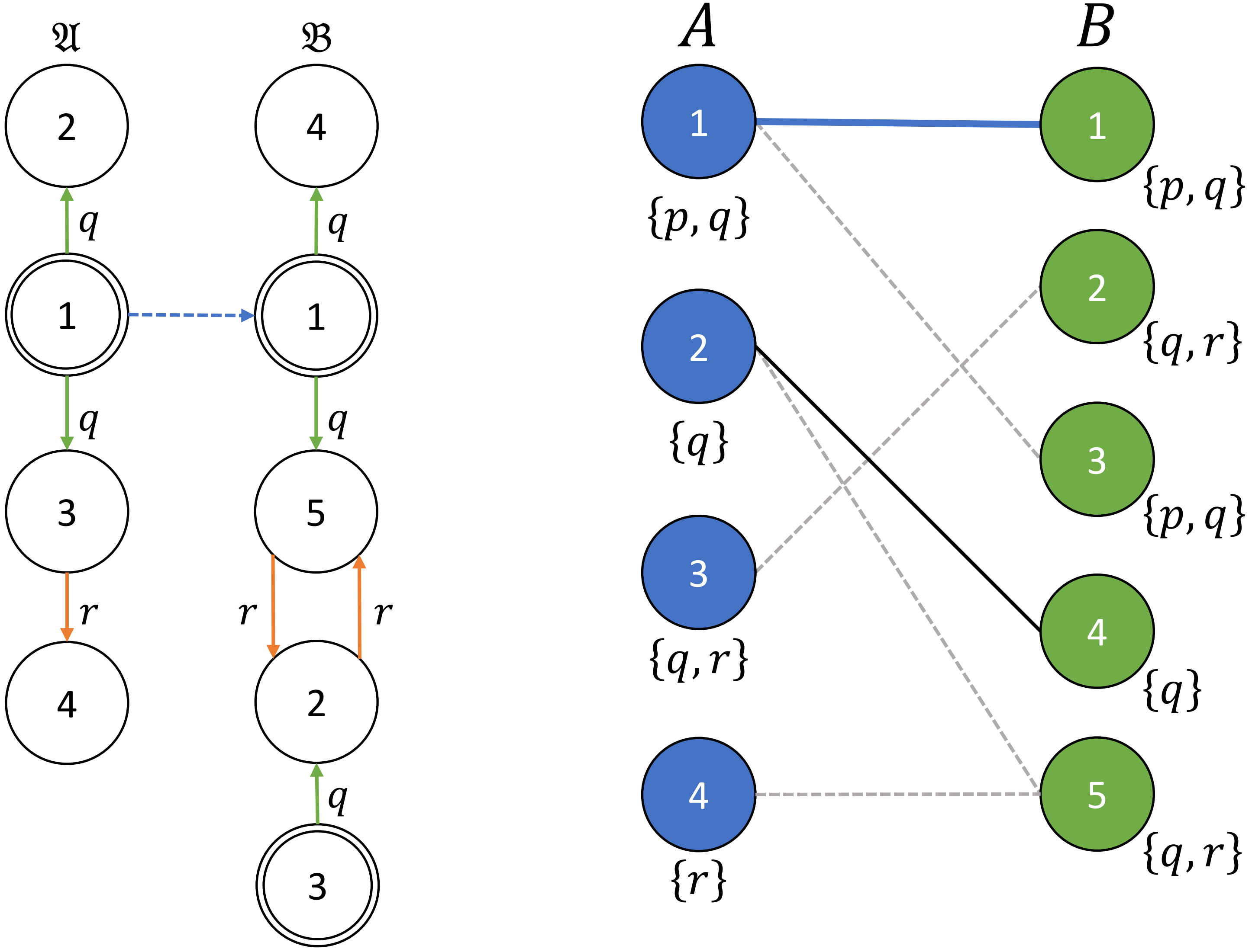
- Reduce search space
  - Remove inconsistent edges
- Preserve embeddings
- Efficiently Computable $O(E^2)$
  - Fixpoint Algorithm[1]

[Russel and Norvig. 2009][1]

Consistency

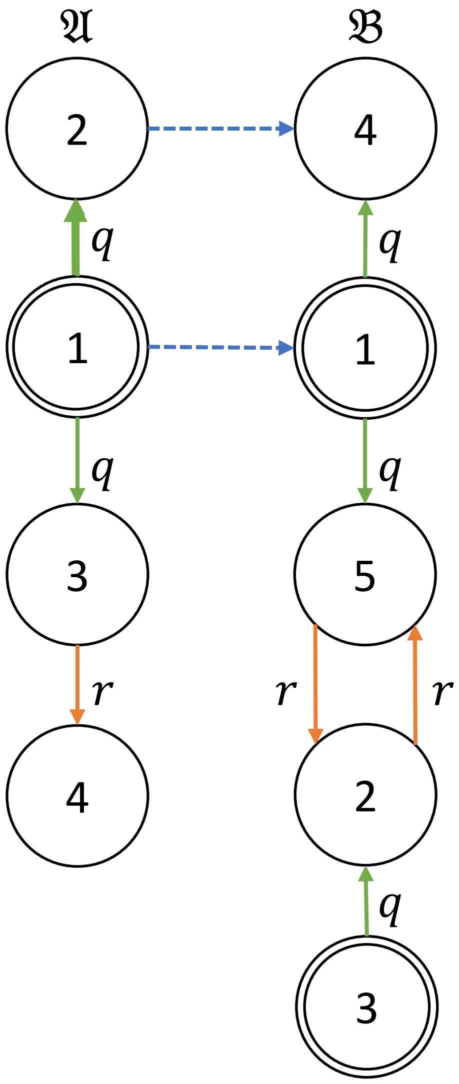


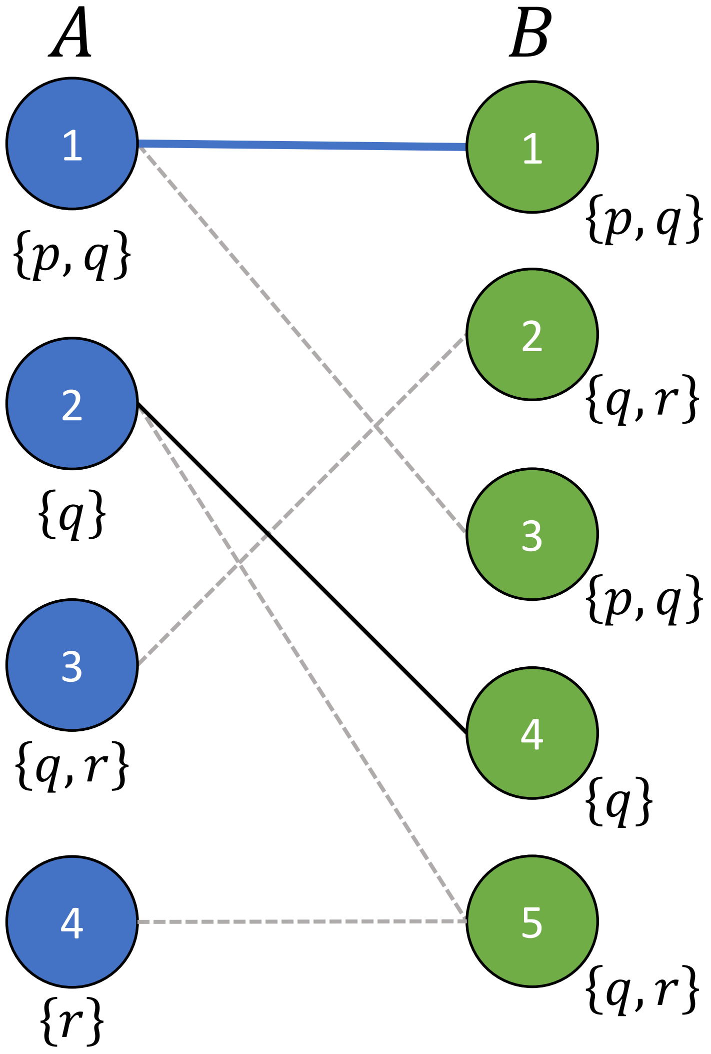
Consistency



Consider edge ⟨1,1⟩:

Consistency

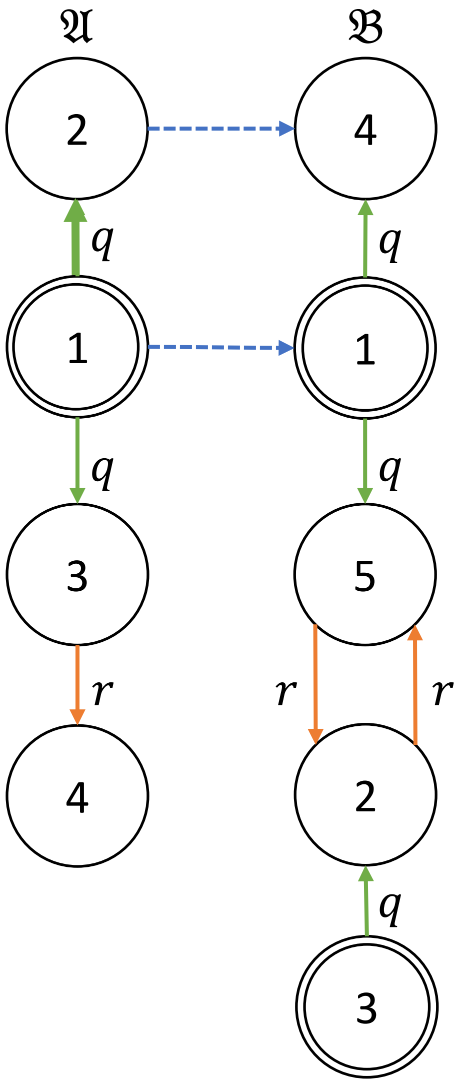


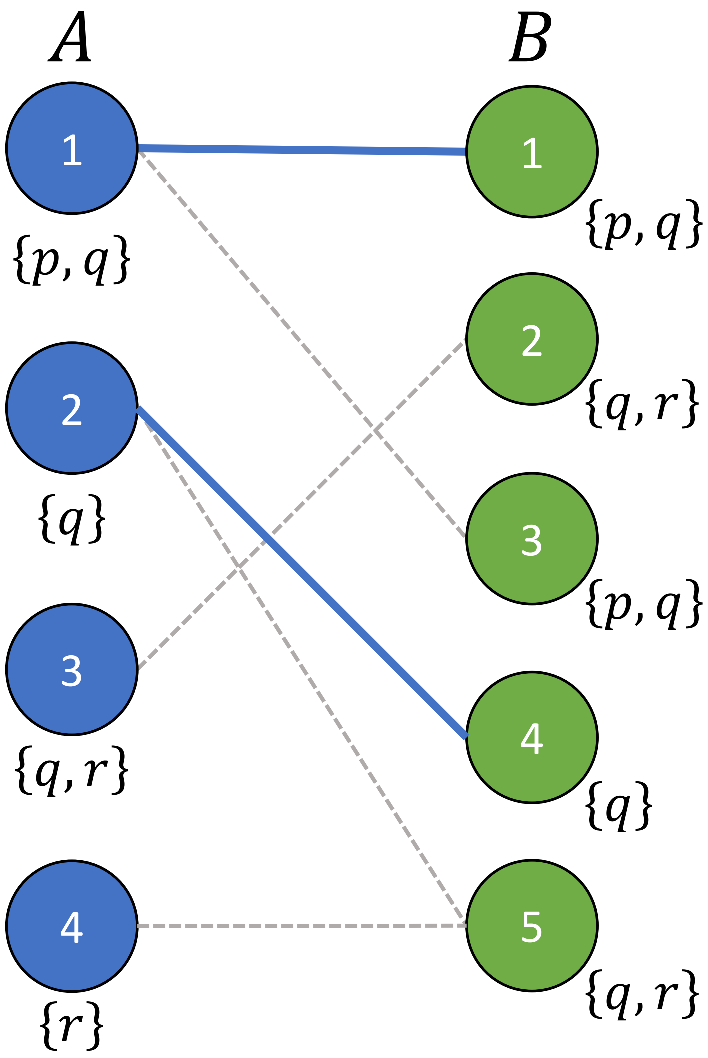


Consider edge ⟨1,1⟩:

- Consider $q(1,2)$

Consistency

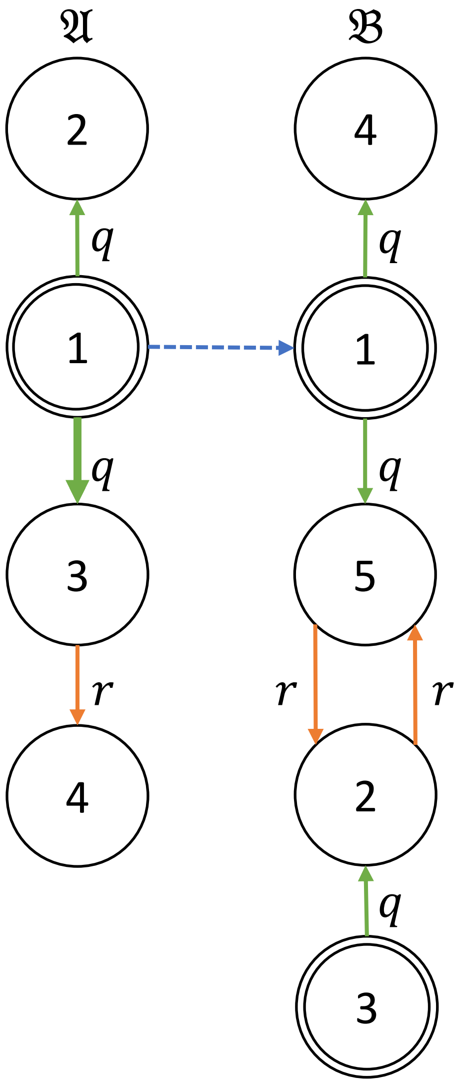


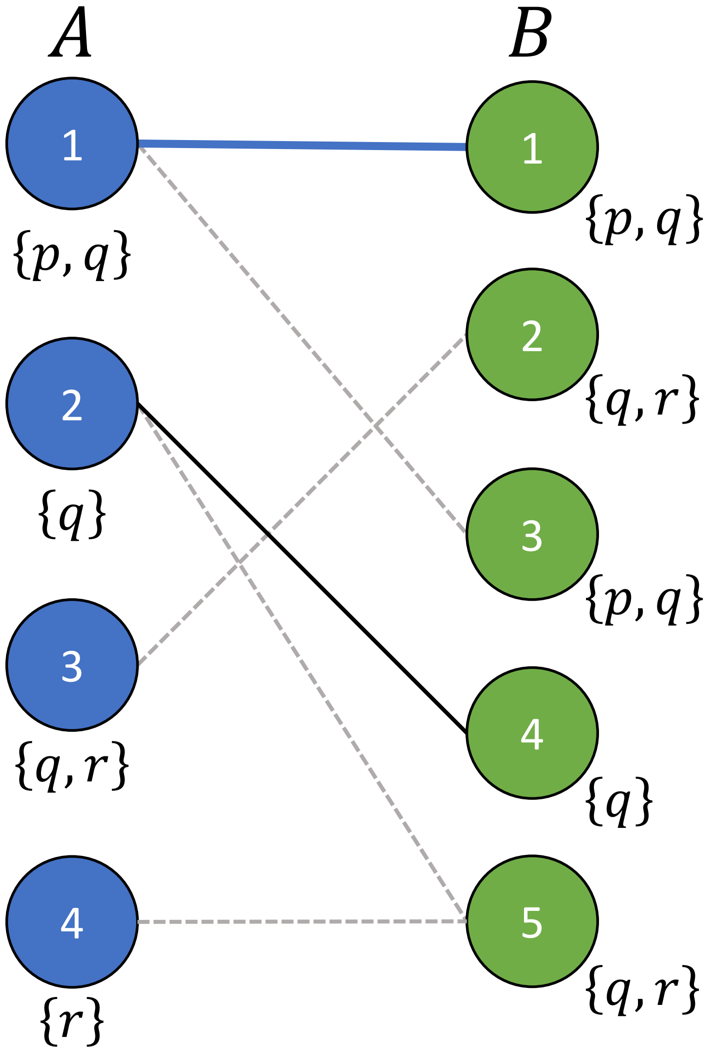


Consider edge $\langle 1,1\rangle$:

- Consider $q(1,2)$
  - $\exists q(1,4) \in \mathfrak{B} \wedge \langle 2,4\rangle \in G$

Consistency

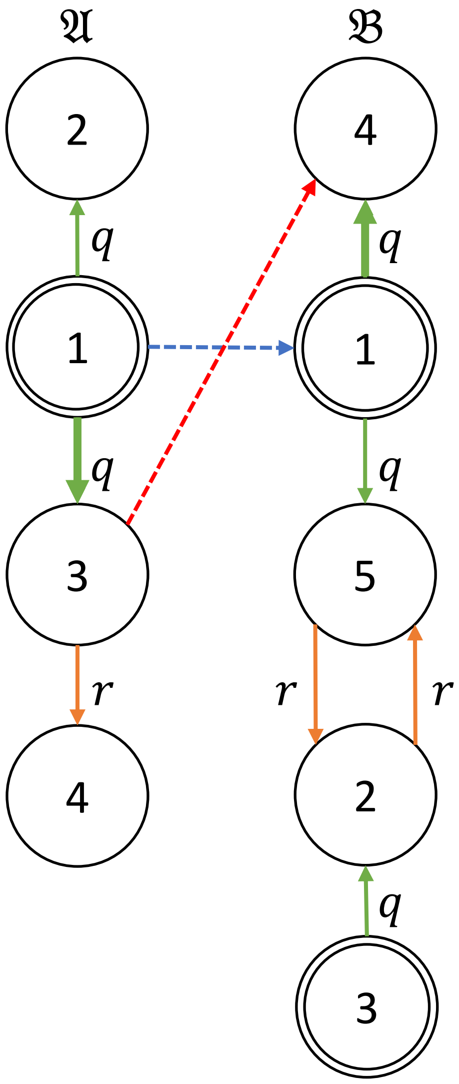


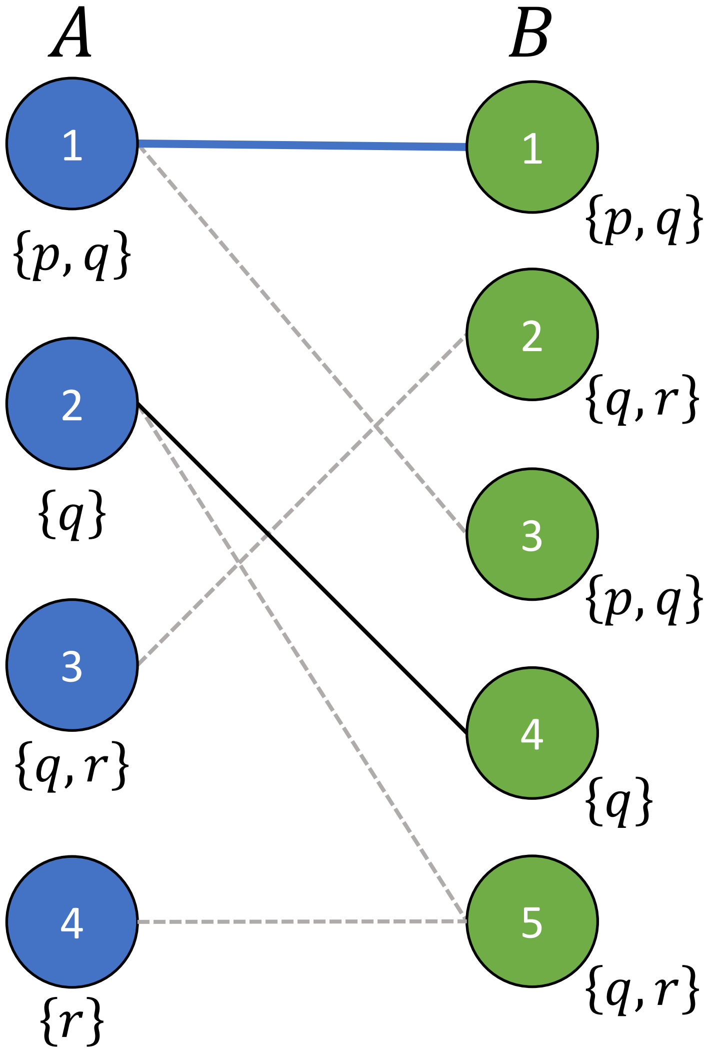


Consider edge $\langle 1,1 \rangle$:

- Consider $q(1,2)$
  - $\exists q(1,4) \in \mathfrak{B} \wedge \langle 2,4 \rangle \in G$
- Consider $q(1,3)$

Consistency

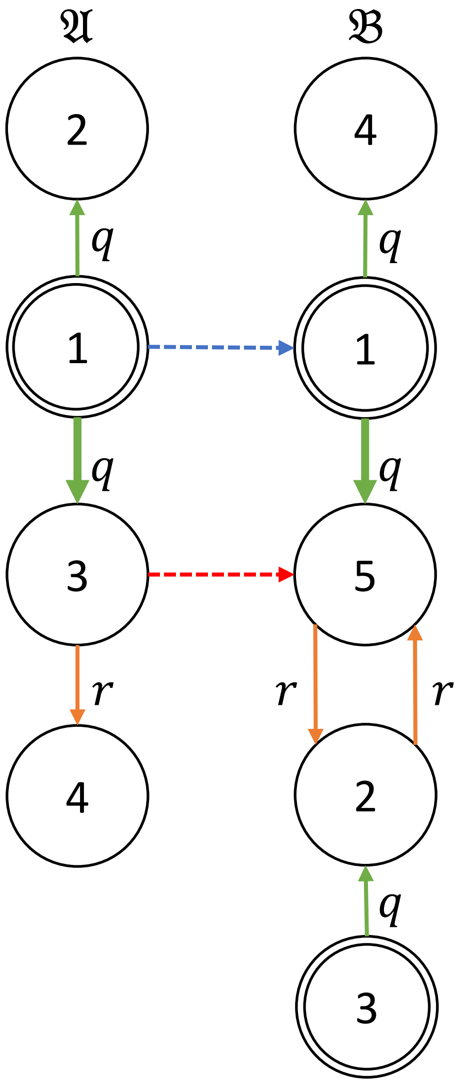


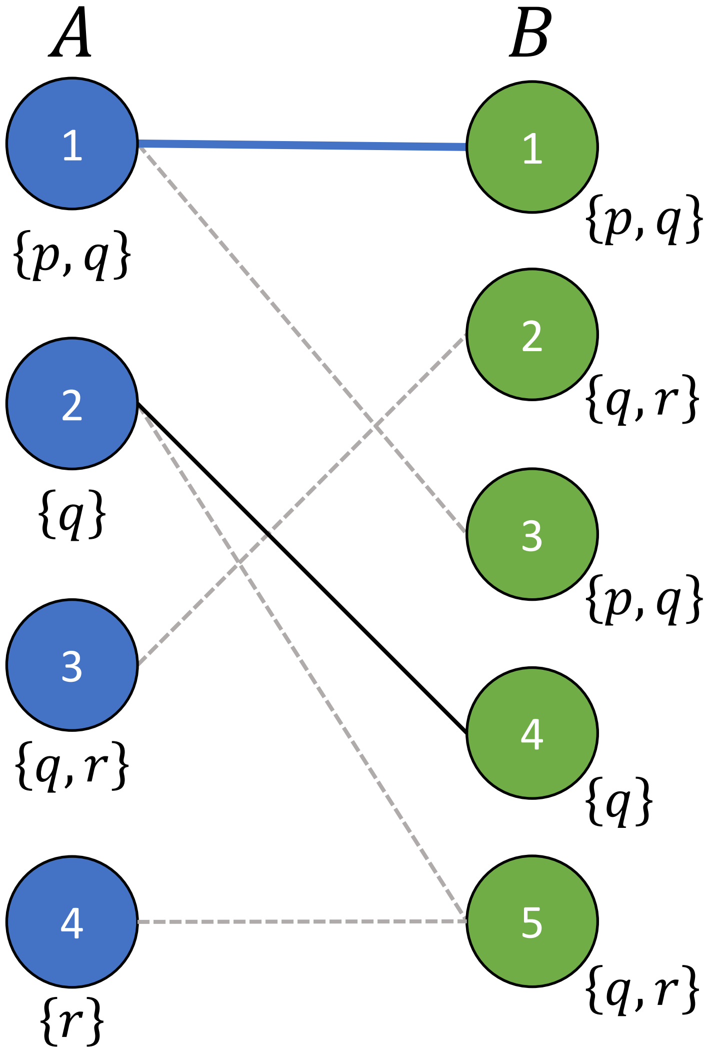


Consider edge $\langle 1,1 \rangle$:

- Consider $q(1,2)$
  - $\exists q(1,4) \in \mathfrak{B} \wedge \langle 2,4 \rangle \in G$
- Consider $q(1,3)$
  - $\exists q(1,4) \in \mathfrak{B}$ but $\langle 3,4 \rangle \notin G$

Consistency

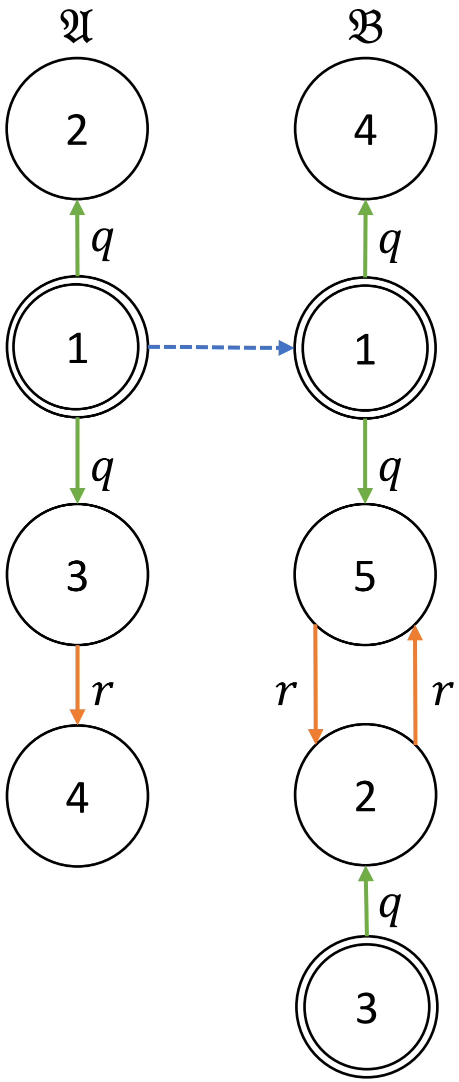


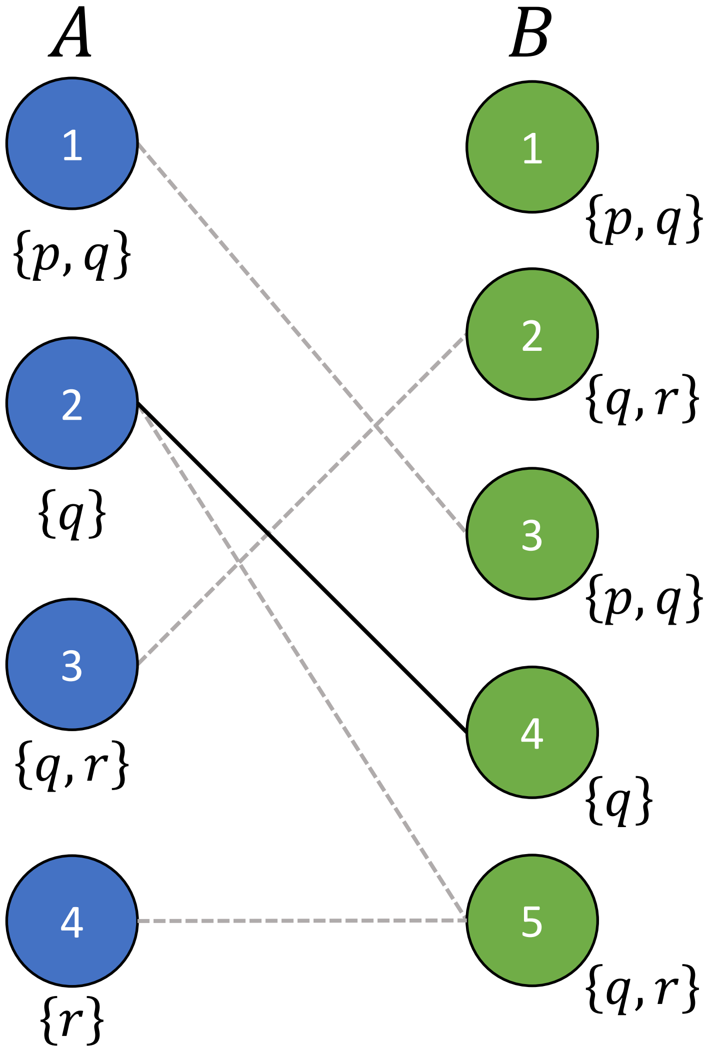


Consider edge $\langle 1,1 \rangle$:

- Consider $q(1,2)$
  - $\exists q(1,4) \in \mathfrak{B} \wedge \langle 2,4 \rangle \in G$
- Consider $q(1,3)$
  - $\exists q(1,4) \in \mathfrak{B}$ but $\langle 3,4 \rangle \notin G$
  - $\exists q(1,5) \in \mathfrak{B}$ but $\langle 3,5 \rangle \notin G$

Consistency





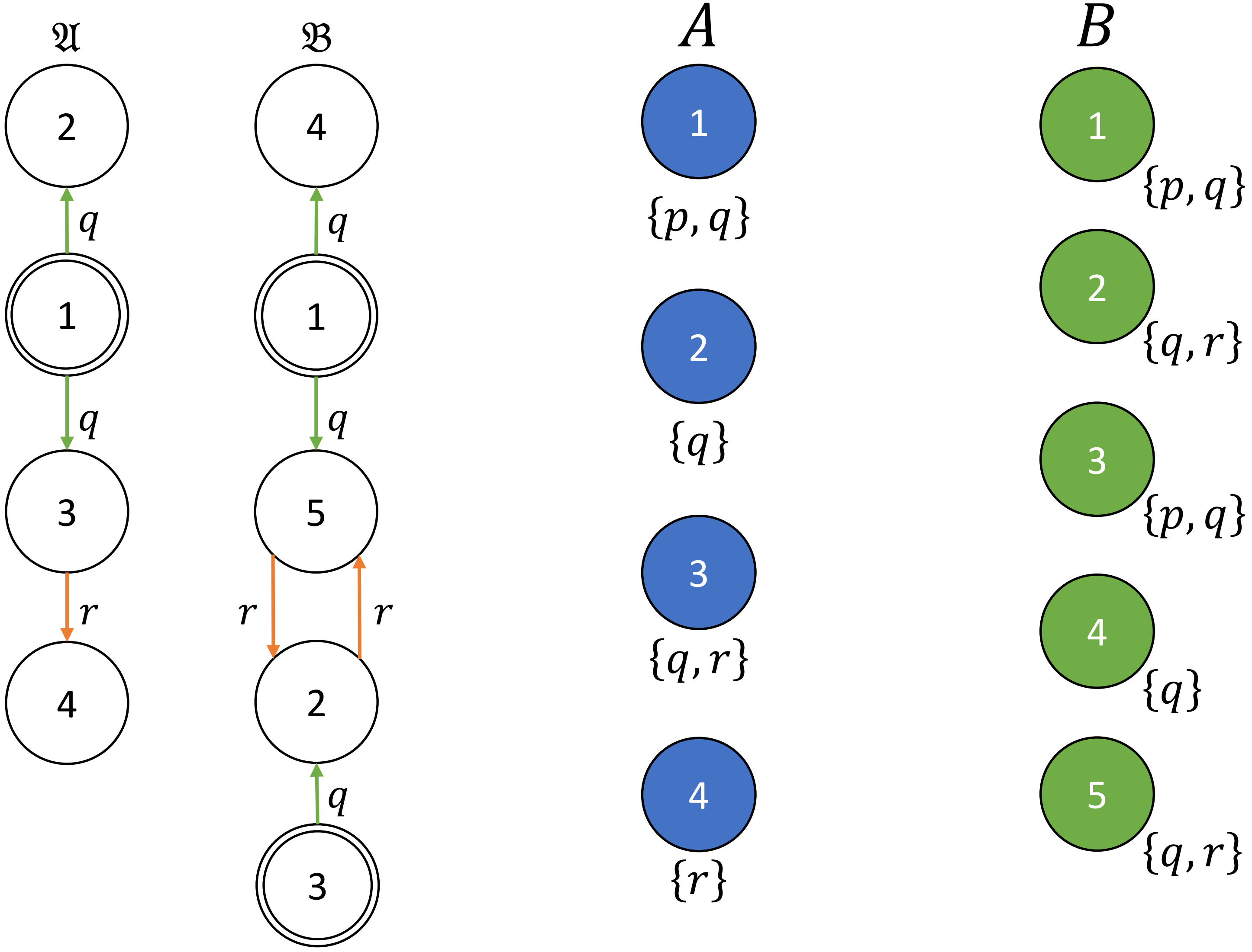
Consider edge $\langle 1,1 \rangle$:

- Consider $q(1,2)$
  - $\exists q(1,4) \in \mathfrak{B} \wedge \langle 2,4 \rangle \in G$
- Consider $q(1,3)$
  - $\exists q(1,4) \in \mathfrak{B}$ but $\langle 3,4 \rangle \notin G$
  - $\exists q(1,5) \in \mathfrak{B}$ but $\langle 3,5 \rangle \notin G$

$\therefore \langle 1,1 \rangle$ is inconsistent

- Remove $\langle 1,1 \rangle$

Consistency



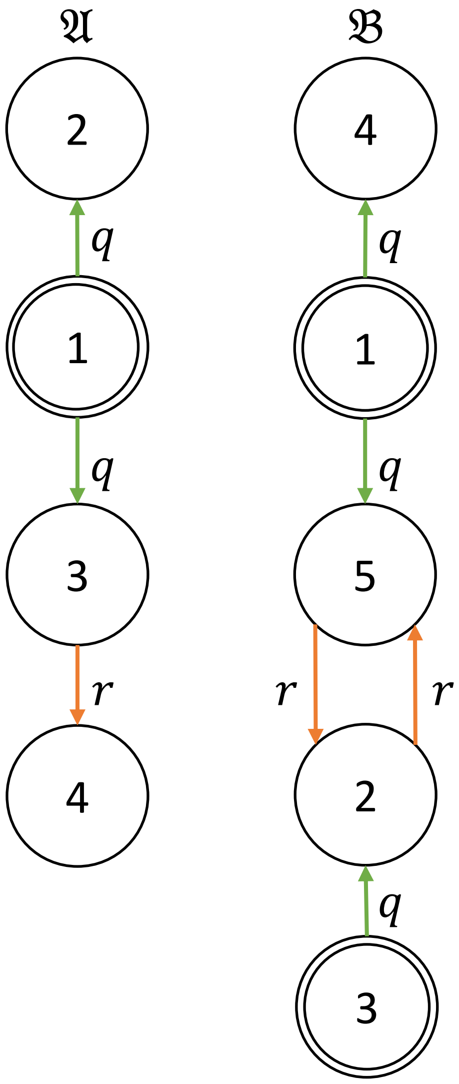
Consider edge $\langle 1,1 \rangle$:

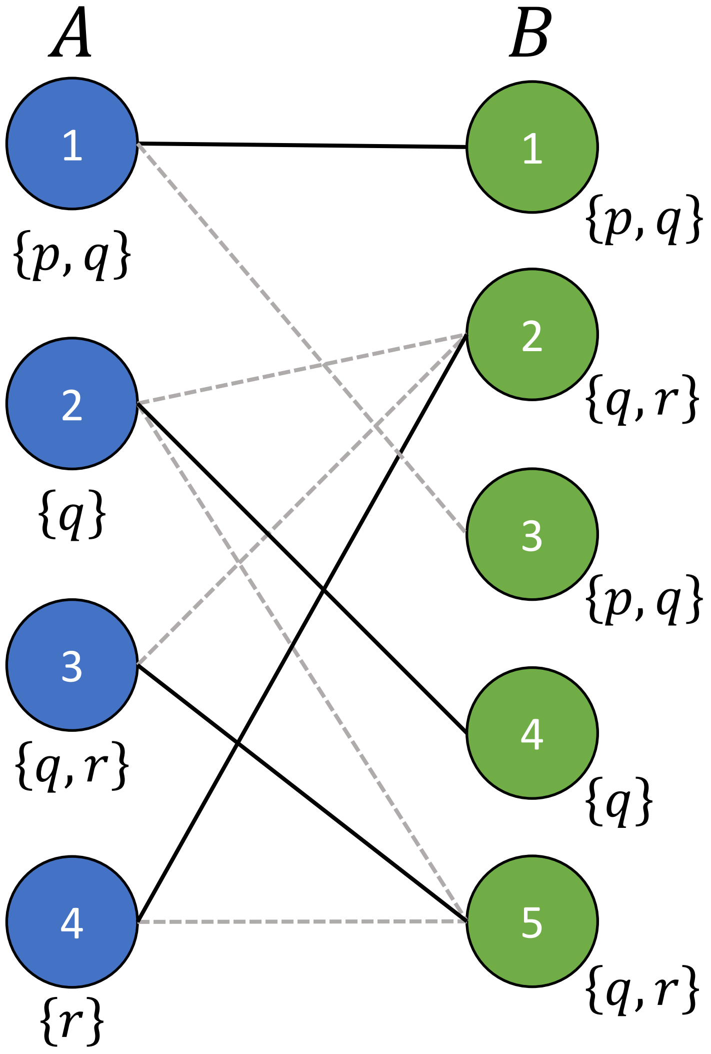
- Consider $q(1,2)$
  - $\exists q(1,4) \in \mathfrak{B} \wedge \langle 2,4 \rangle \in G$
- Consider $q(1,3)$
  - $\exists q(1,4) \in \mathfrak{B}$ but $\langle 3,4 \rangle \notin G$
  - $\exists q(1,5) \in \mathfrak{B}$ but $\langle 3,5 \rangle \notin G$

$\therefore \langle 1,1 \rangle$ is inconsistent

  - Remove $\langle 1,1 \rangle$
- Repeat until consistent

MatchEmbeds

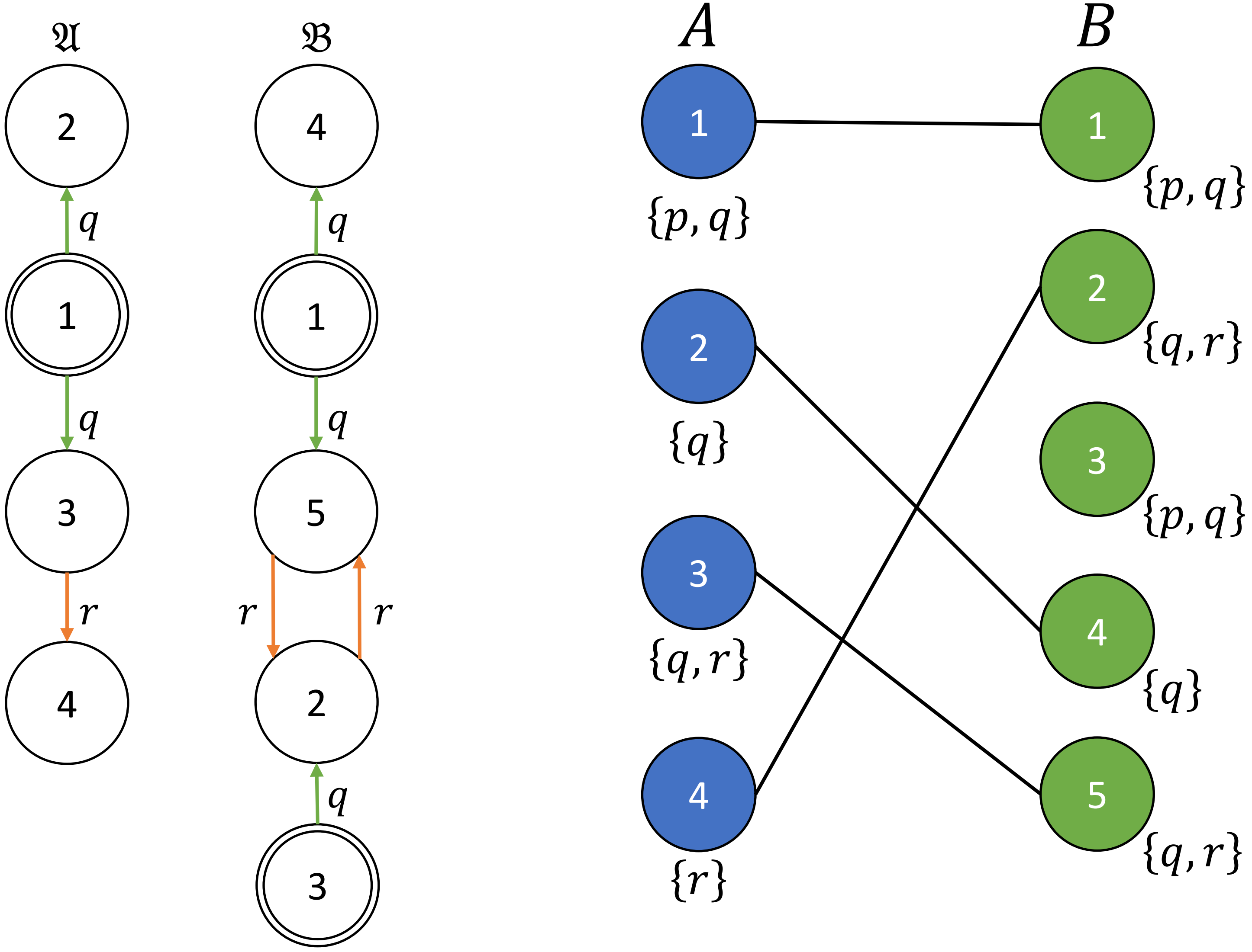




Backtrack $[3 \longmapsto 2]$

- Blame $\langle 3,2 \rangle$

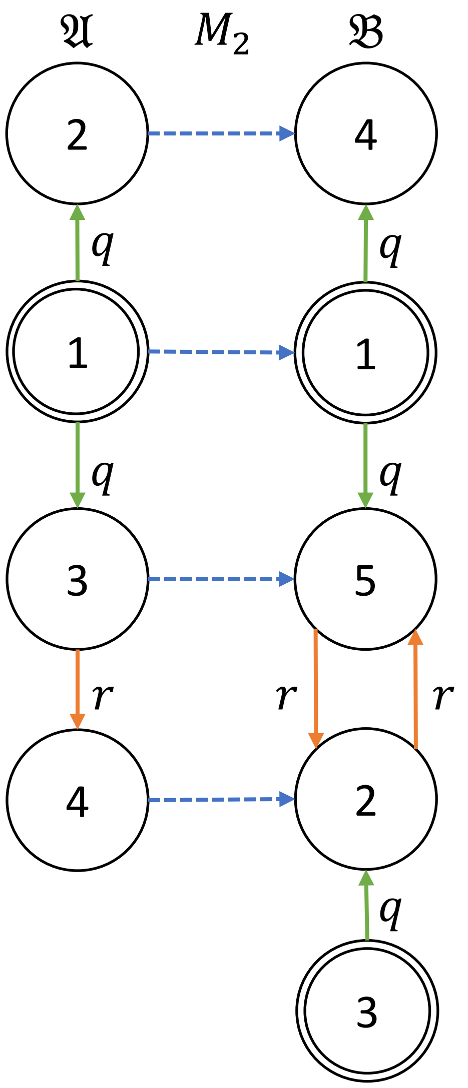
MatchEmbeds

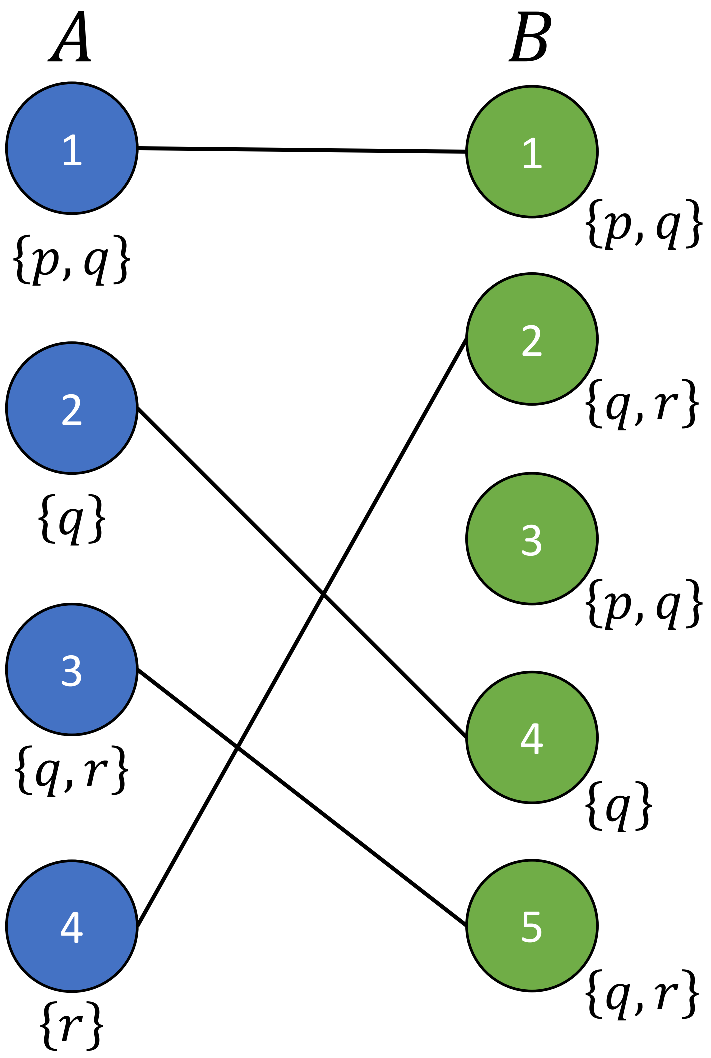


Backtrack $[3 \longmapsto 2]$

- Blame $\langle 3{,}2 \rangle$
- Compute consistent sub-graph

MatchEmbeds

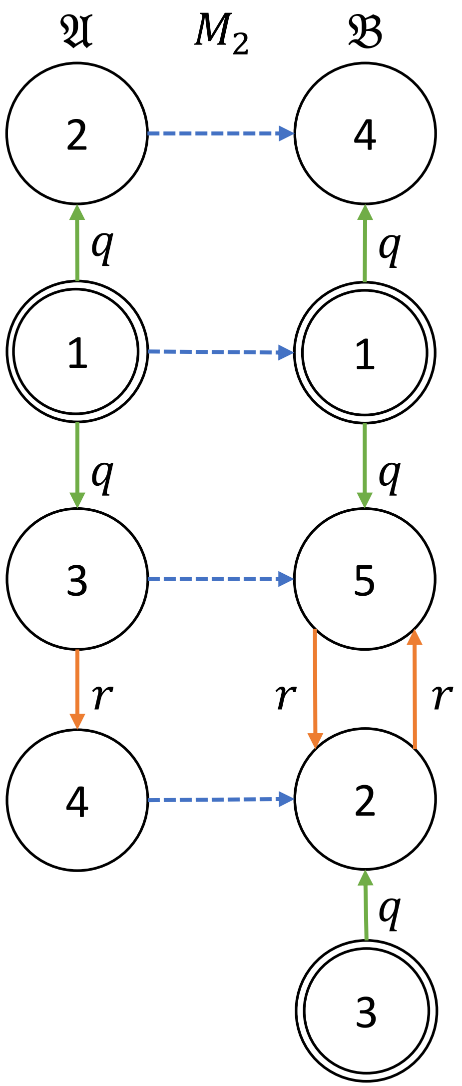


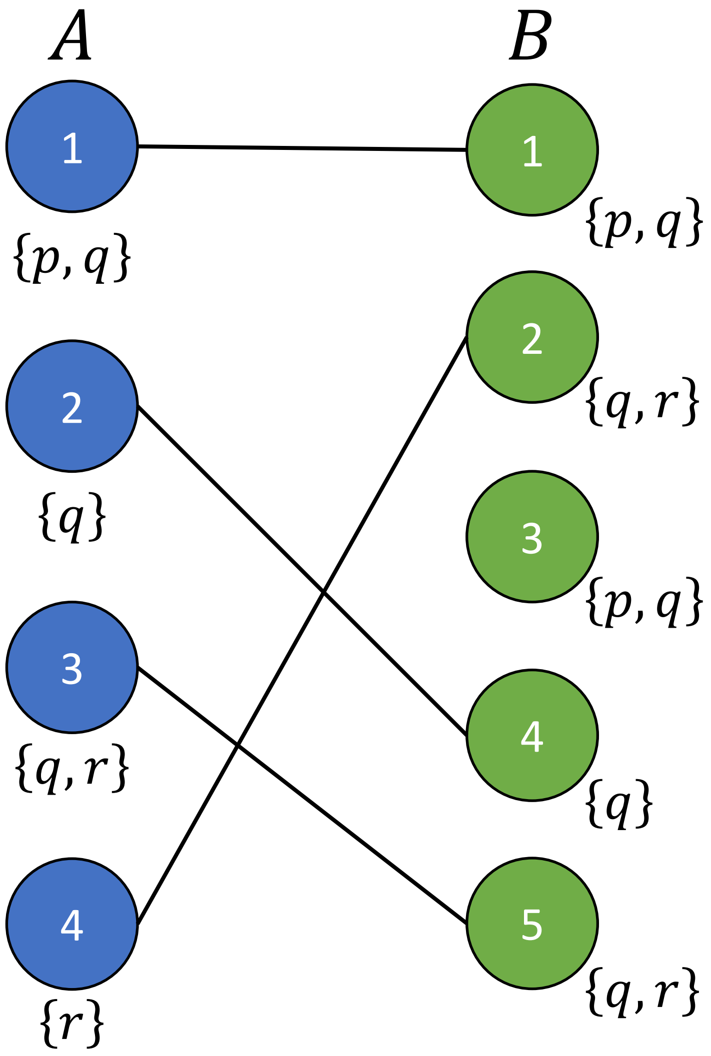


Compute Matching

$M_2 \overset{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

MatchEmbeds





Compute Conflict Set

$M_2 \stackrel{\text{def}}{=} \{\langle 1,1\rangle, \langle 2,4\rangle, \langle 3,5\rangle, \langle 4,2\rangle\}$

$Conflict(M_2) \stackrel{\text{def}}{=} \emptyset$

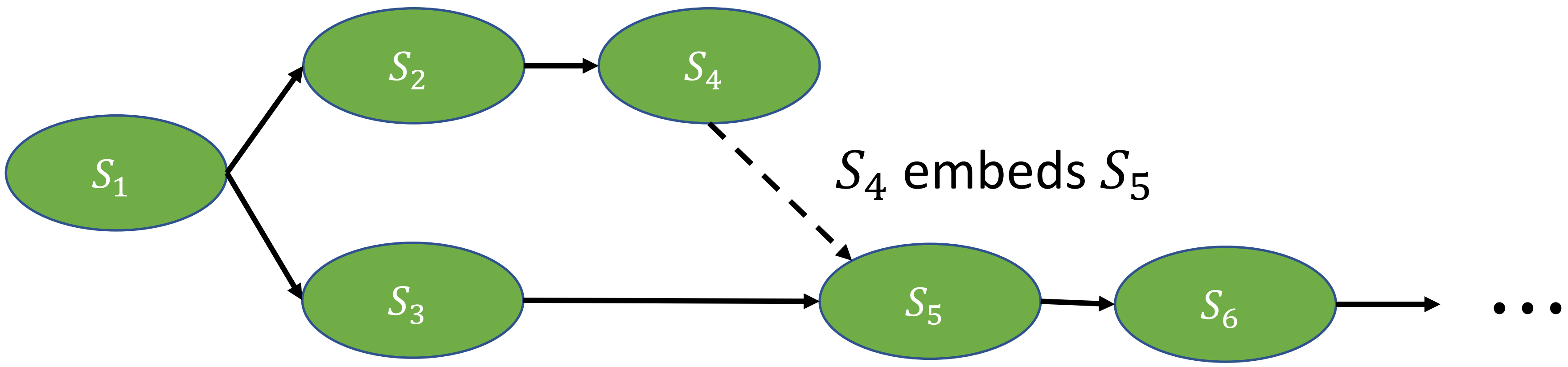
M_2 is an Embedding

MatchEmbeds Algorithm

Function embeds(G)
  $G \leftarrow filter(G)$
  $M \leftarrow$ **maximum_matching**(G)
  if $|M| \neq |G.A|$ **then**
    return false
  end
  if f_M *is an embedding* **then**
    return true
  end
  Select a decision $\langle a, b \rangle \in M$
  if embeds$(G \setminus \{\langle u, v \rangle \in E : u = a$ **xor** $v = b\})$ **then**
    return true
  else
    return embeds$(G \setminus \{\langle a, b \rangle\})$
  end

MatchEmbeds for Program verification

- Used for pruning state space exploration of parameterized programs
  - Parameterized program states abstracted as structures
  - Check if the current state is subsumed by a previously explored state
  - Can we do better than a brute force search?



Multi-Source Single-Target Embeddings

- Check if some $\mathfrak{A} \in str$ in a set of structures embeds into $\mathfrak{B}$
- Key idea: no need to check all such structures
  - Map each $\mathfrak{A}$ to $v(\mathfrak{A}) \in \mathbb{N}^d$ (for some d)
    - Crucial property: if $\mathfrak{A}$ embeds into $\mathfrak{B}$ then $v(\mathfrak{A}) \leq v(\mathfrak{B})$
  - Store structures in a k-d tree
  - Use range queries on k-d tree and test returned structures

Experiments

- Is MatchEmbeds Practical?
  - Compared to CSP, SAT, and Graph Isomorphism Solvers:

CSP	SAT	Subgraph Isomorphism
Gecode[1] HaifaCSP[2] OrTools[3]	Lingeling[4] Cryptominisat[5]	VF2[6] Glasgow[7]

[Schulte et al. 2018][1] [Veksler and Strichman. 2015][2] [Perron. 1011][3] [Biere. 2013][4]
[Soos and Nohl. 2010][5] [Cordella et al. 2004][6] [McCreesh and Prosser. 2015][7]

Experiments

- Is MatchEmbeds Practical?
  - Compared to CSP, SAT, and Graph Isomorphism Solvers: …
  - Does it improve performance of our client model checker (proof-of-concept implementation of *Proof Spaces*[1])?
  - Does the k-d tree of structures improve *Proof Spaces*?
- Can MatchEmbeds solve difficult problem instances?

[Farzan et al. 2016][1]

Experiment Count Threads

```
main():
  count = 0
  for i = 1 to N:
    fork thread
  assert(count ≤ N)

thread():
  count = count+1
```

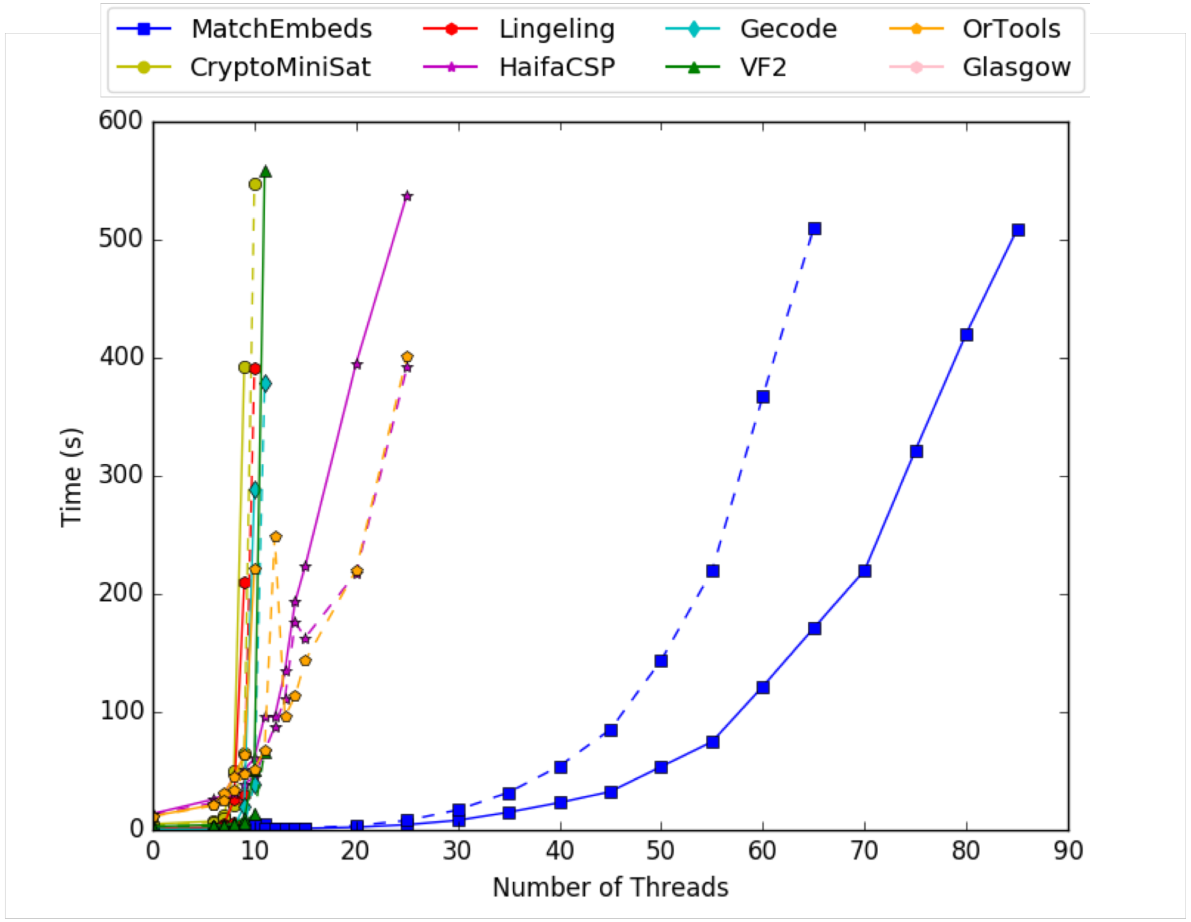
Cactus Plot:

x-axis: # of threads

y-axis: Time to verify

Using k-d Tree (solid line)

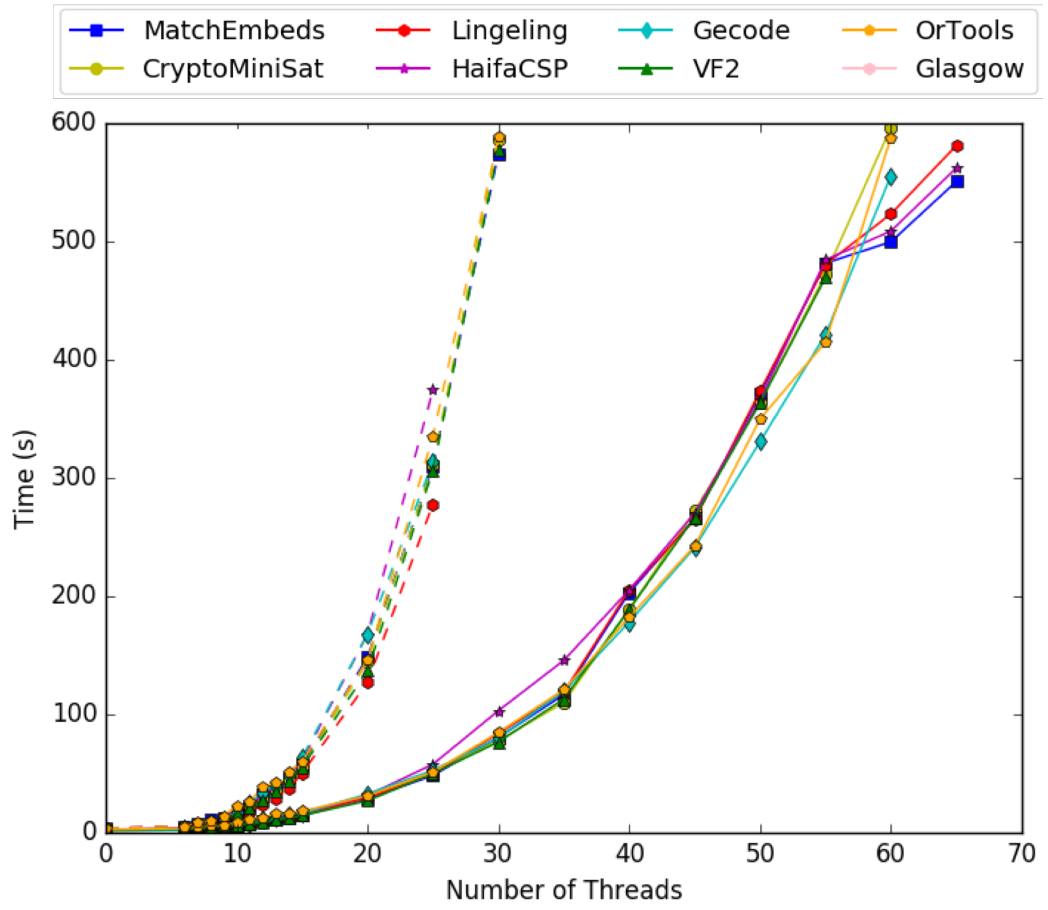
Brute-force (dashed line)



Experiment Secret Sharing

```
main():
 from = 0
 while (*)
  local secret = *
  assume(secret > 0)
  for i = 1 to N:
   to = secret
   fork thread
   while (to > 0): skip
  if (from > 0):
   assert(from == secret)

thread():
  local m = to
  to = 0
  from = m
```



Experiments

- Is MatchEmbeds Practical?
  - Compared to CSP, SAT, and Graph Isomorphism Solvers: …
  - Does it improve performance of our client model checker (proof-of-concept implementation of *Proof Spaces*[1])?
  - Does the k-d structure improve *Proof Spaces*?
- Can MatchEmbeds solve difficult problem instances?

[Farzan et al. 2016][1]

Experiment Random Structures

- Our previous model-checking examples lead to “easy” instances
- Generate random structures
  - Hard instances are *hard* to find[1]
  - Generalize method for hard subgraph isomorphisms[2] to structure embeddings

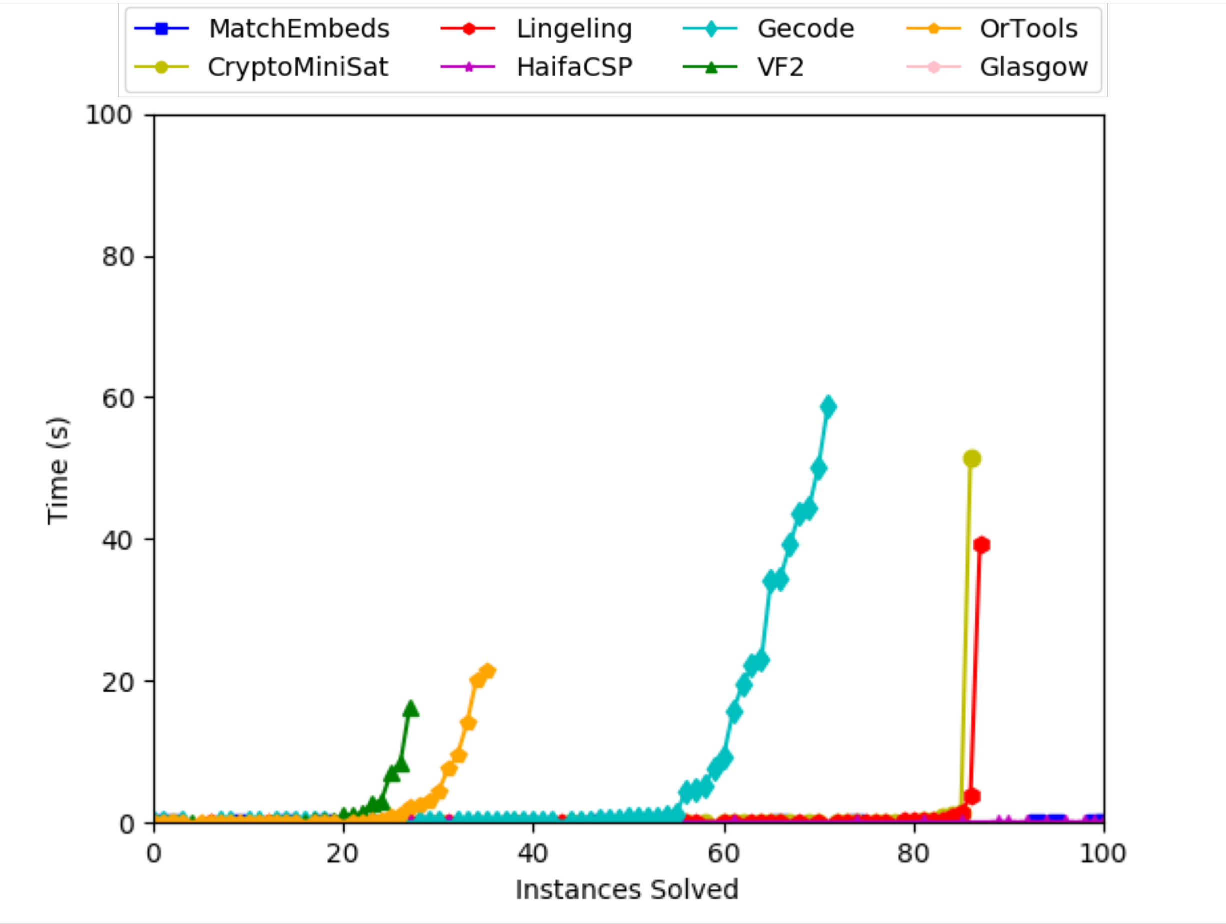
[Cheeseman et al. 1991][1] [McCreesh et al. 2016][2]

Experiment Random Monadic Structures

- $|A| = 40, |B| = 50$
- 3 monadic predicates
- Generate 100 instances
  - 53 positive embeddings
  - 47 negative embeddings

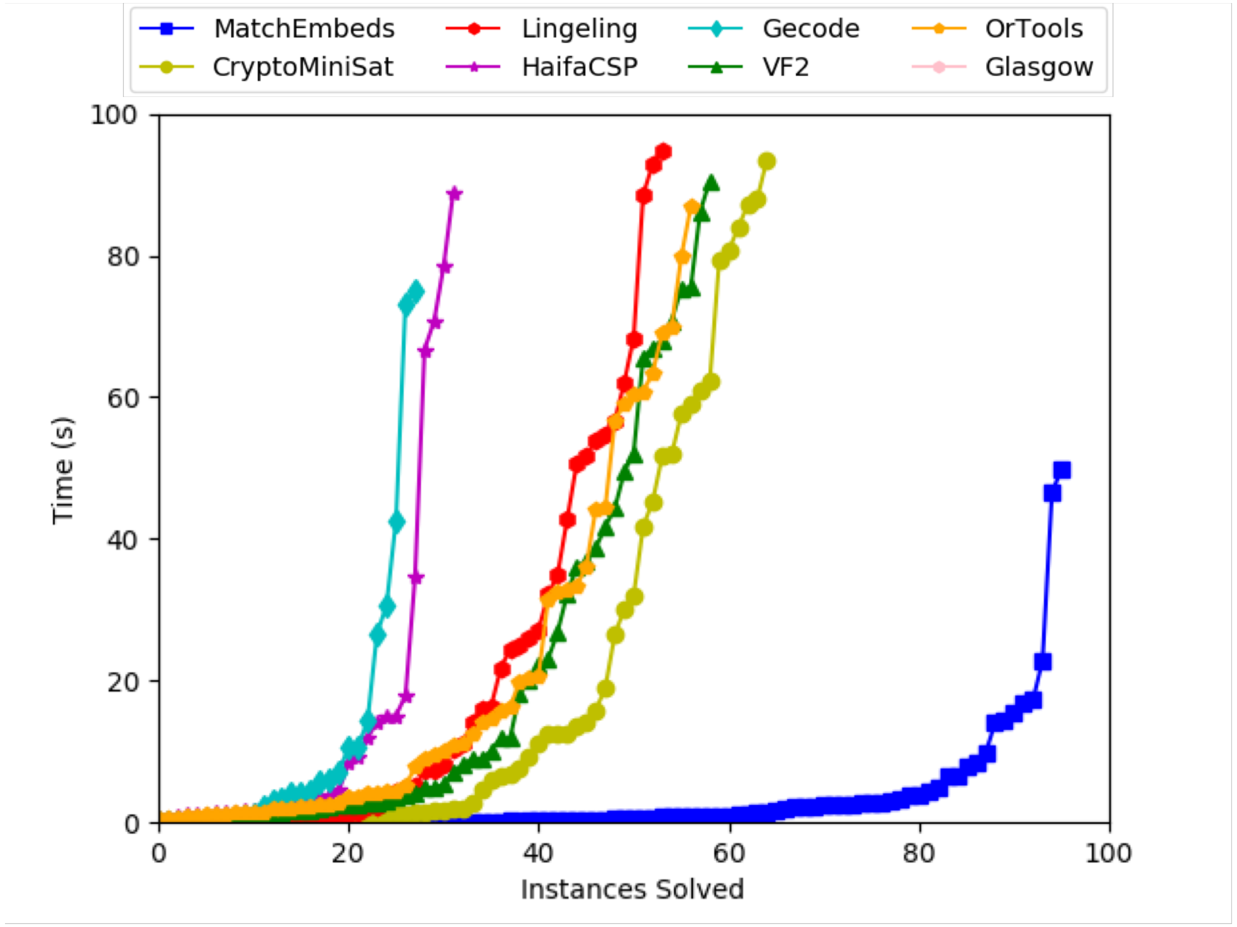
- MatchEmbeds & HaifaCSP[1]
  - Polytime monadic instances

[Régin. 1994][1]



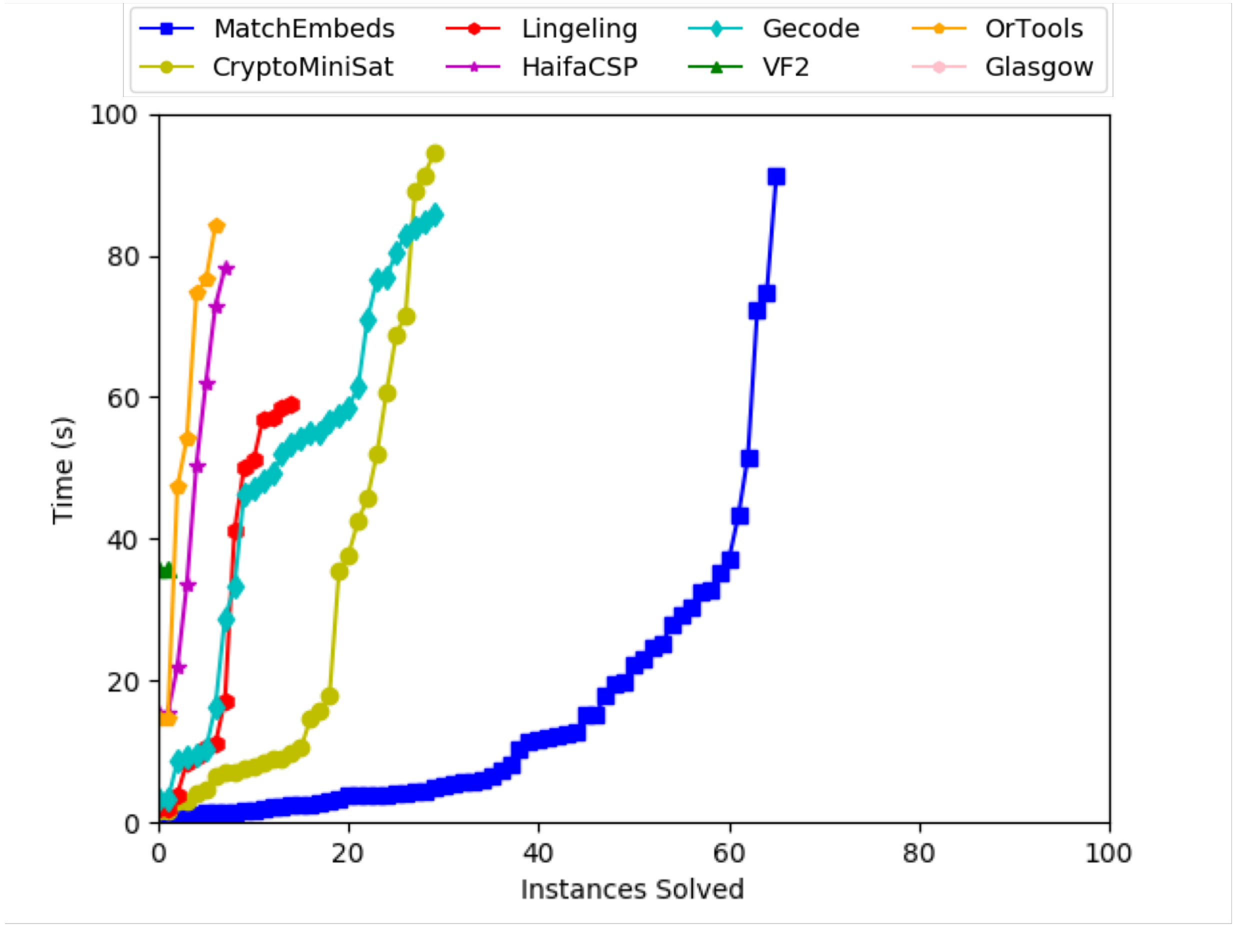
Experiment Random Binary Structures

- $|A| = 20, |B| = 30$
- 3 monadic, 3 binary predicates
- 100 Instances
  - 46 positive embeddings
  - 49 negative embeddings
  - 5 unsolved embeddings



Experiment Random Ternary Structures

- $|A| = 10, |B| = 30$
- 3 monadic, 3 binary, and 3 ternary predicates
- 100 Instances
  - 35 positive embeddings
  - 32 negative embeddings
  - 33 unsolved embeddings



Summary

- MatchEmbeds: a practical algorithm for the structure embedding problem
  - Polytime for monadic instances
  - 1-2 orders of magnitude faster than SMT/CSP/Graph-Isomorphism
  - Proof space clients scale to ~twice as many threads
- Key ideas:
  - Search over the space of total matchings in a bipartite graph
  - Speed up single-source, multi-target queries using k-d trees

References

[1] Kincaid, Z. Podelski, A., Farzan, A. *Proof Spaces for Unbounded Parallelism*. POPL, pgs. 407-420 (2015).
[2] Finkel, A. Schnoebelen, Ph. *Well Structured Transition Systems Everywhere.* Theoretical Computer Science Vol 256:1, pgs. 63-92 (2001).
[3] Hopcroft, J., Karp, R. *An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs*. SIAM Journal of Computing, Vol. 2, No. 5 : pgs. 225-231 (1973).
[4] Régin, J.C.: *A filtering Algorithm for Constraints of Difference in CSPs*. In: AAAI. pgs. 362-367 (1994)
[5] Russell, S.J., Norvig, P. *Artificial Intelligence - a Modern Approach*, 3rd Edition. Prentice Hall series in Artificial Intelligence. Prentice Hall (2009)
[6] McCreesh, C., Prosser, P., Trimble, J.: Heuristics and really hard instances for subgraph isomorphism problems. In: IJCAI. pp. 631{638 (2016)
[7] Cheeseman, P.C., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: IJCAI. vol. 91, pp. 331{340 (1991)

[8] Schulte, C., Lagerkvist, M., Tack, G.: GECODE - An open, free, ecient constraint solving toolkit. www.gecode.org
[9] Veksler, M., Strichman, O.: Learning general constraints in CSP. In: CPAIOR. pp. 410-426 (2015), http://strichman.net.technion.ac.il/haifacsp/
[10] Perron, L.: Operations research and constraint programming at google. In: CP. pp. 2-2 (2011), https://developers.google.com/optimization/
[11] Biere, A.: Lingeling, plingeling and treengeling entering the sat competition 2013. In: Balint, A., Belov, A., Heule, M.J., Järvisalo, M. (eds.) SAT Competition 2013. pp. 51{52. Department of Computer Science Series of Publications B (2013)
[12] Soos, M., Nohl, K., Castelluccia, C.: Cryptominisat. SAT Race solver descriptions (2010).
[13] P. Cordella, L., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. Pattern Anal. Mach. Intell. 26(10), 1367{1372 (Oct 2004)
[14] McCreesh, C., Prosser, P.: A parallel, backjumpig subgraph isomorphism algorithm using supplemetal graphs. In: International conference on principles and practice of constraint programming. pp. 295{312. Springer (2015)