universität freiburg

eSLIM: Using Exact Synthesis for Circuit Minimization

Franz-Xaver Reichl University of Freiburg
Friedrich Slivovsky University of Liverpool
Stefan Szeider TU Wien

IWLS 2025 Programming Contest

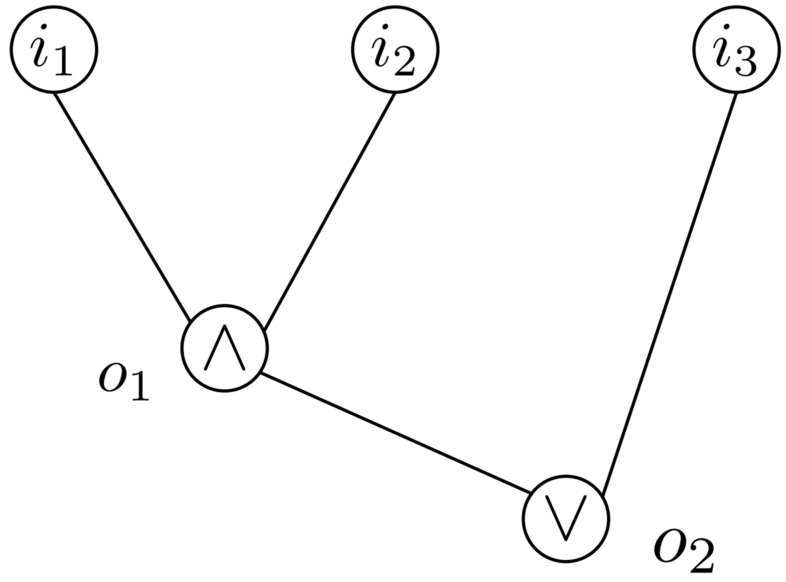
Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 508508079

Exact Synthesis

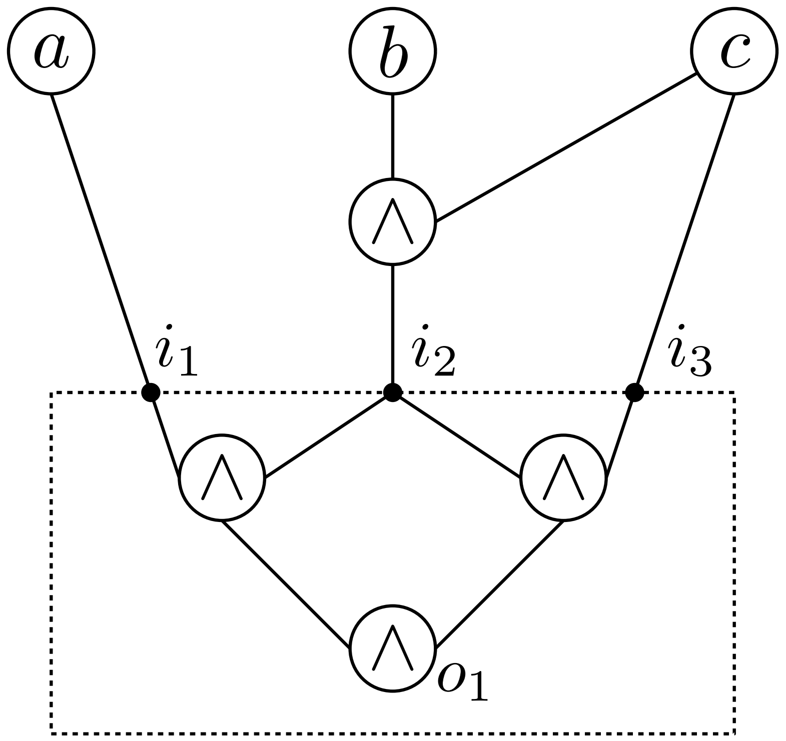
i_1	i_2	i_3	o_1	o_2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

Exact Synthesis

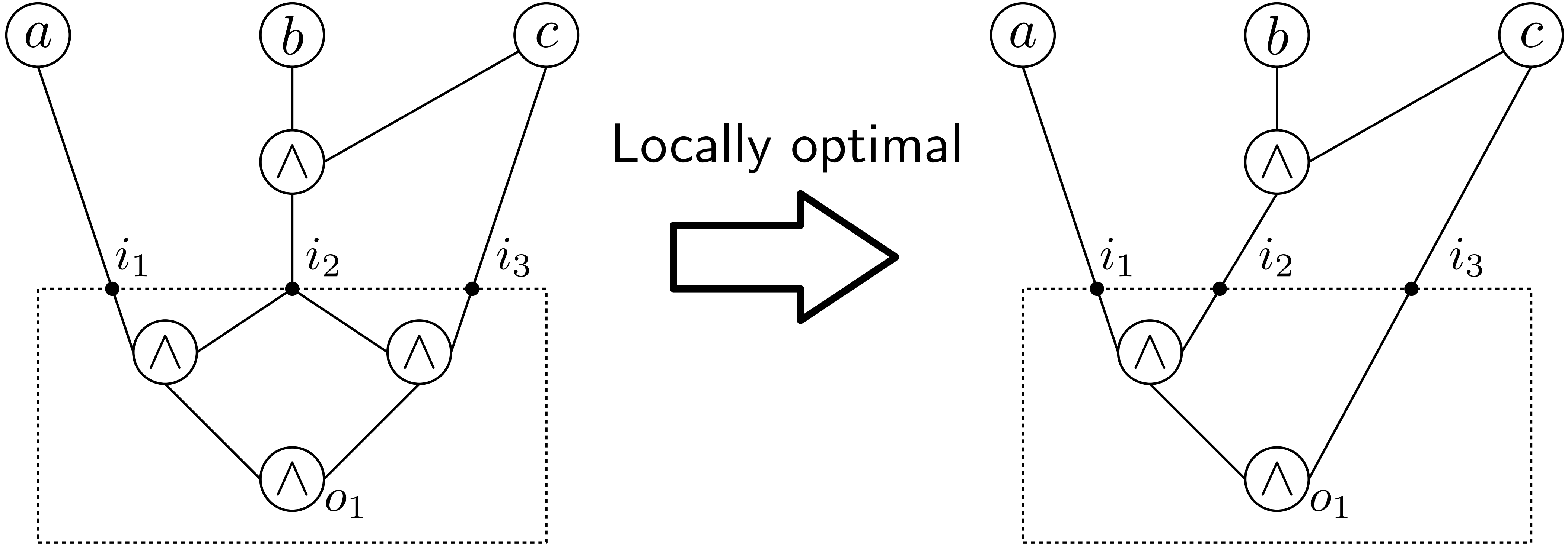
i_1	i_2	i_3	o_1	o_2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1



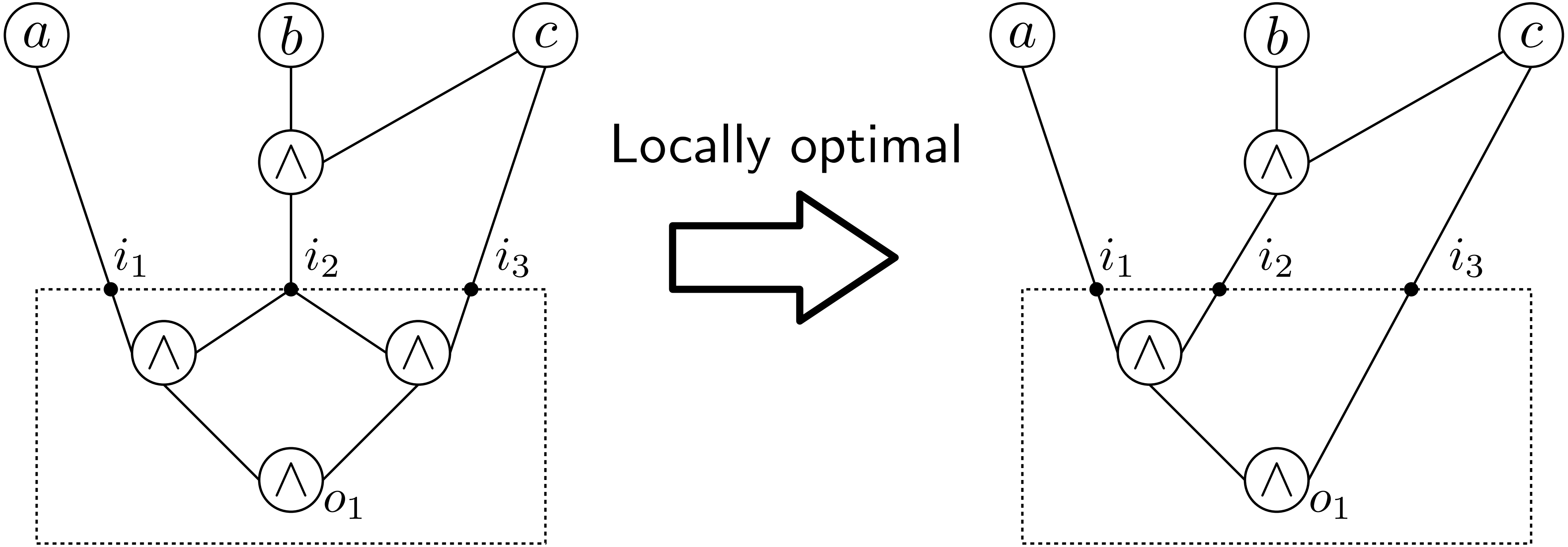
Exact Synthesis of Subcircuits



Exact Synthesis of Subcircuits

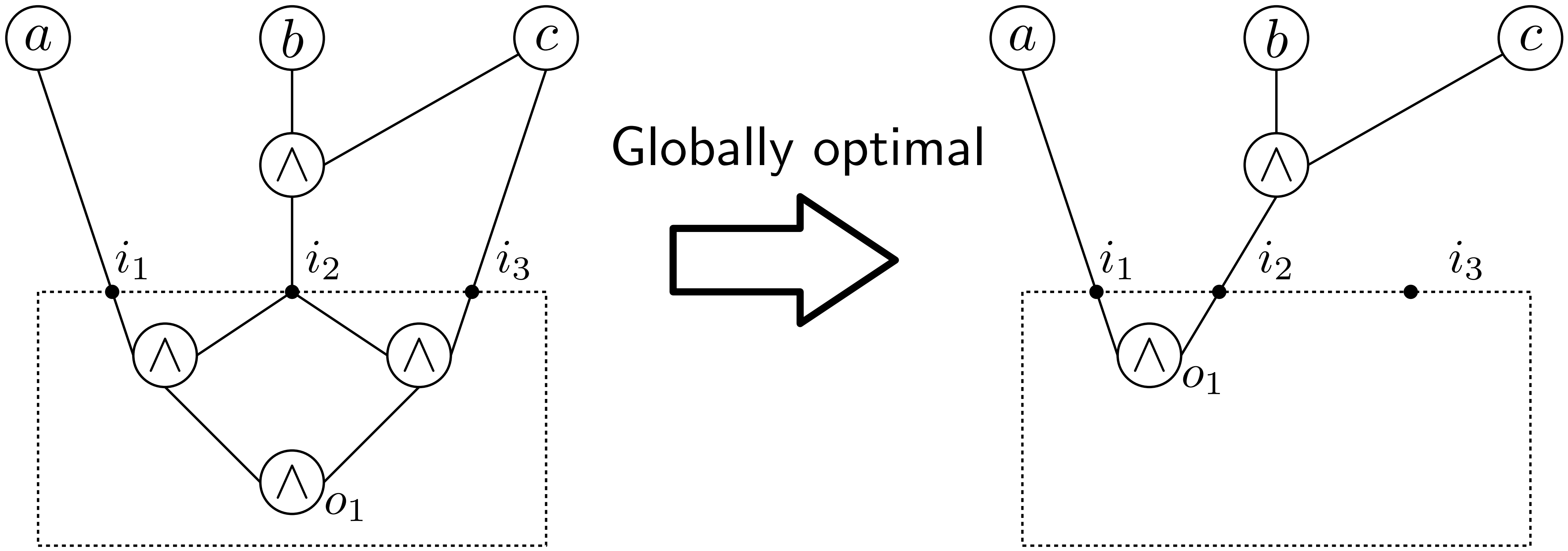


Exact Synthesis of Subcircuits



Not limited to equivalent replacements
$\rightarrow$ Make use of don't cares

Exact Synthesis of Subcircuits

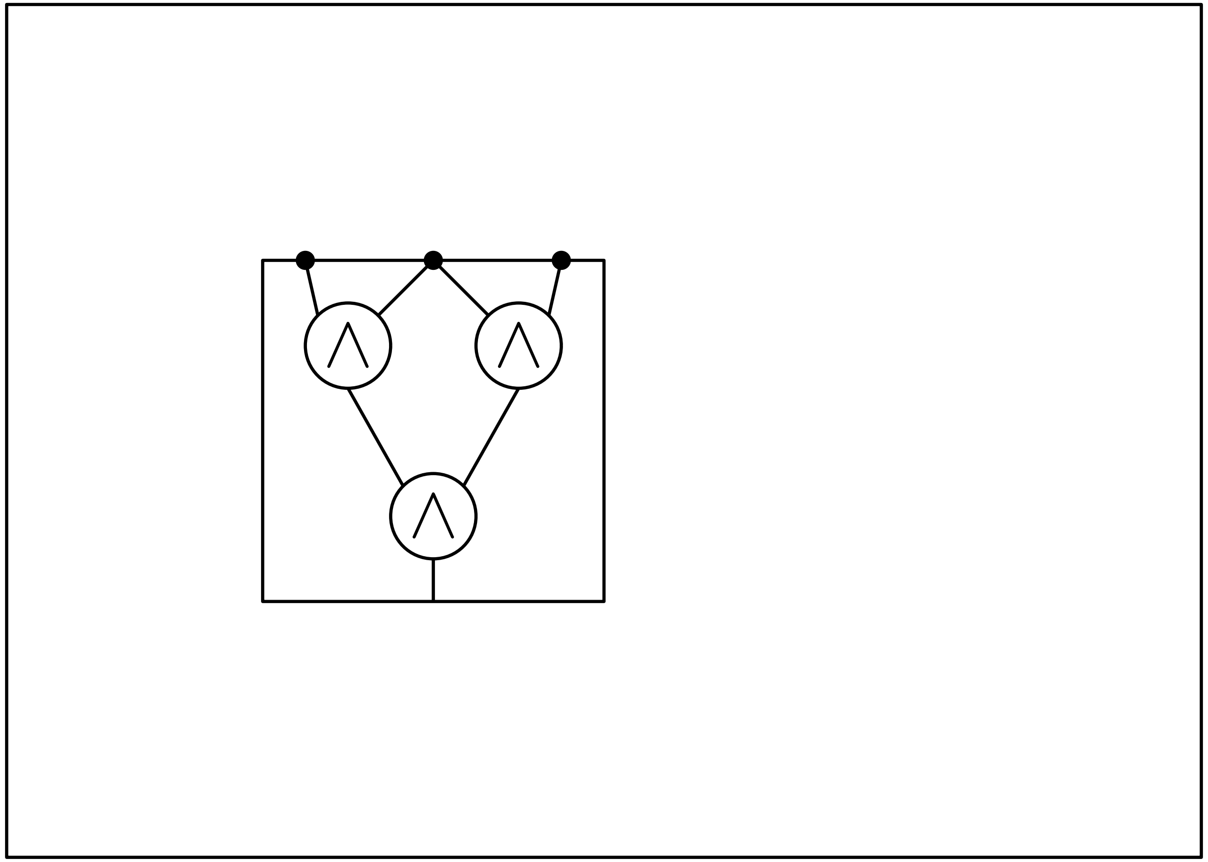


Not limited to equivalent replacements
$\rightarrow$ Make use of don't cares

Workflow

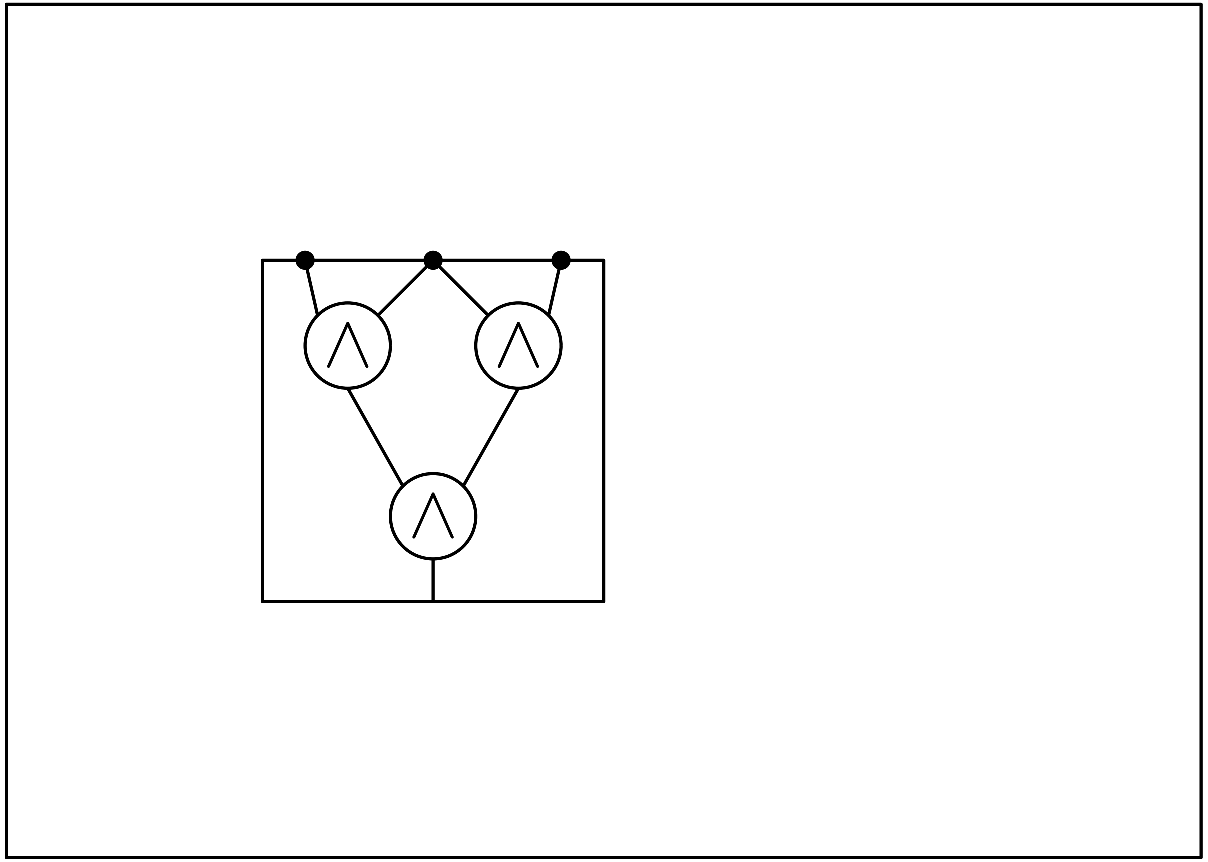
1. Initial circuit

Workflow

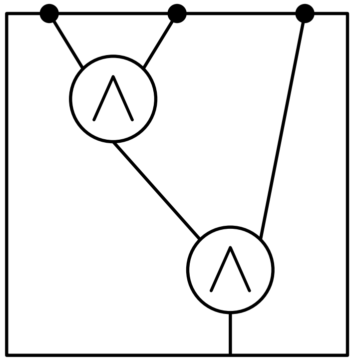


1. Initial circuit
2. Select subcircuit

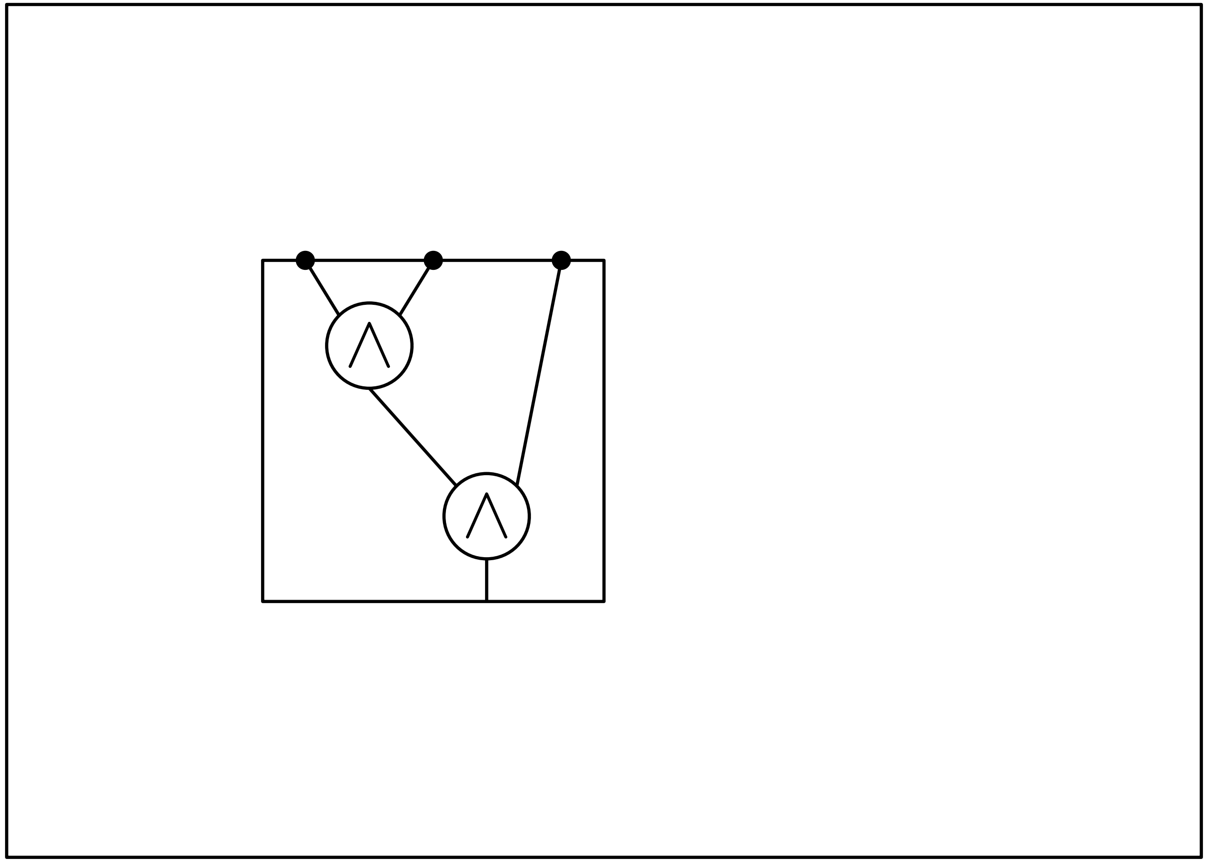
Workflow



1. Initial circuit
2. Select subcircuit
3. Synthesise replacement

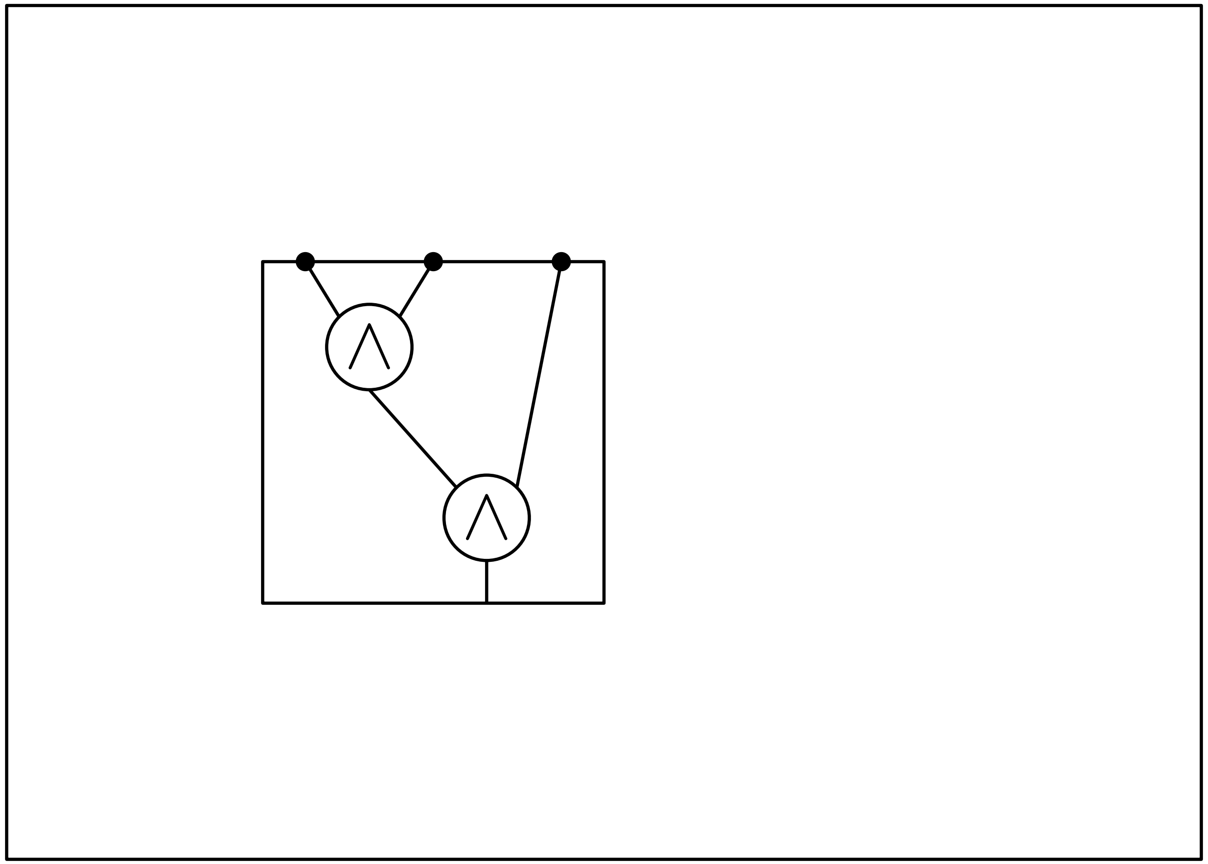


Workflow



1. Initial circuit
2. Select subcircuit
3. Synthesise replacement
4. Insert Replacement

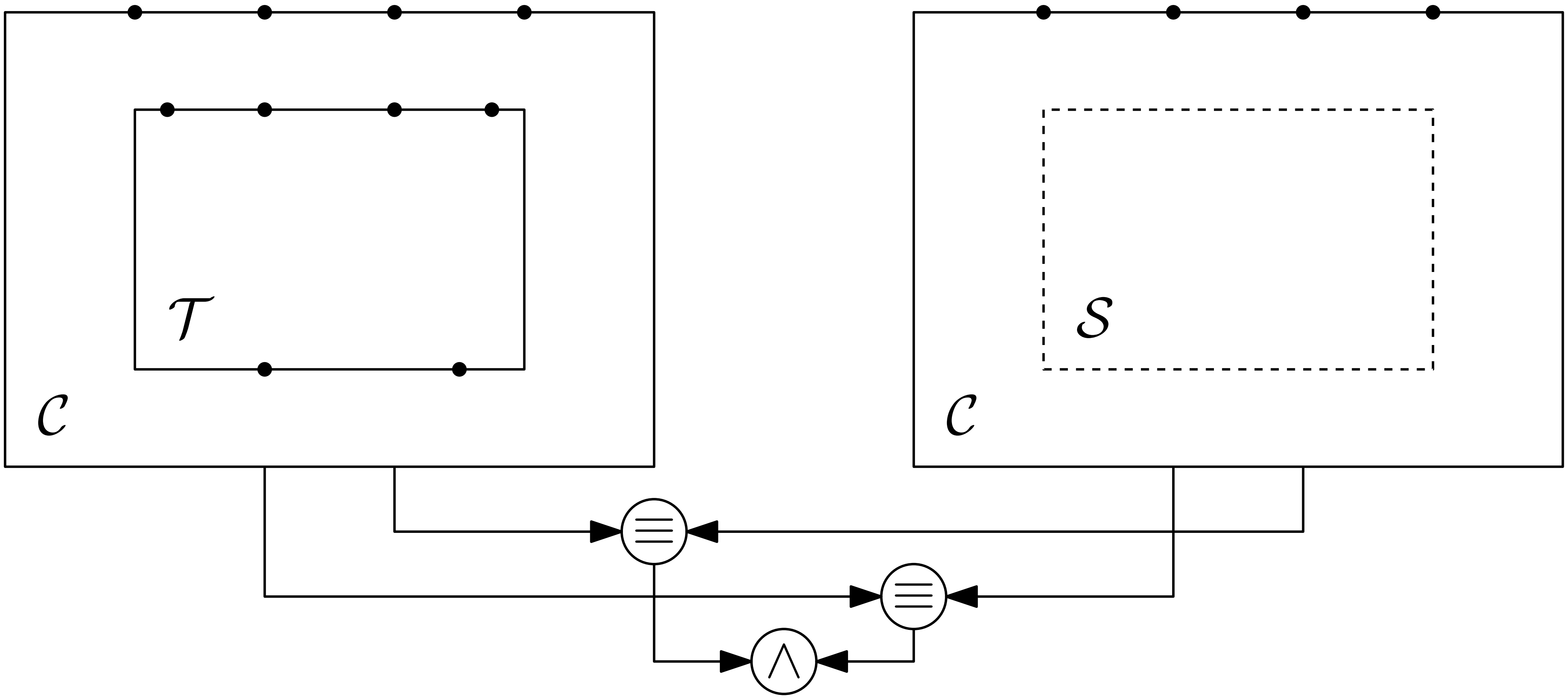
Workflow



1. Initial circuit
2. Select subcircuit
3. Synthesise replacement
4. Insert Replacement
5. Repeat

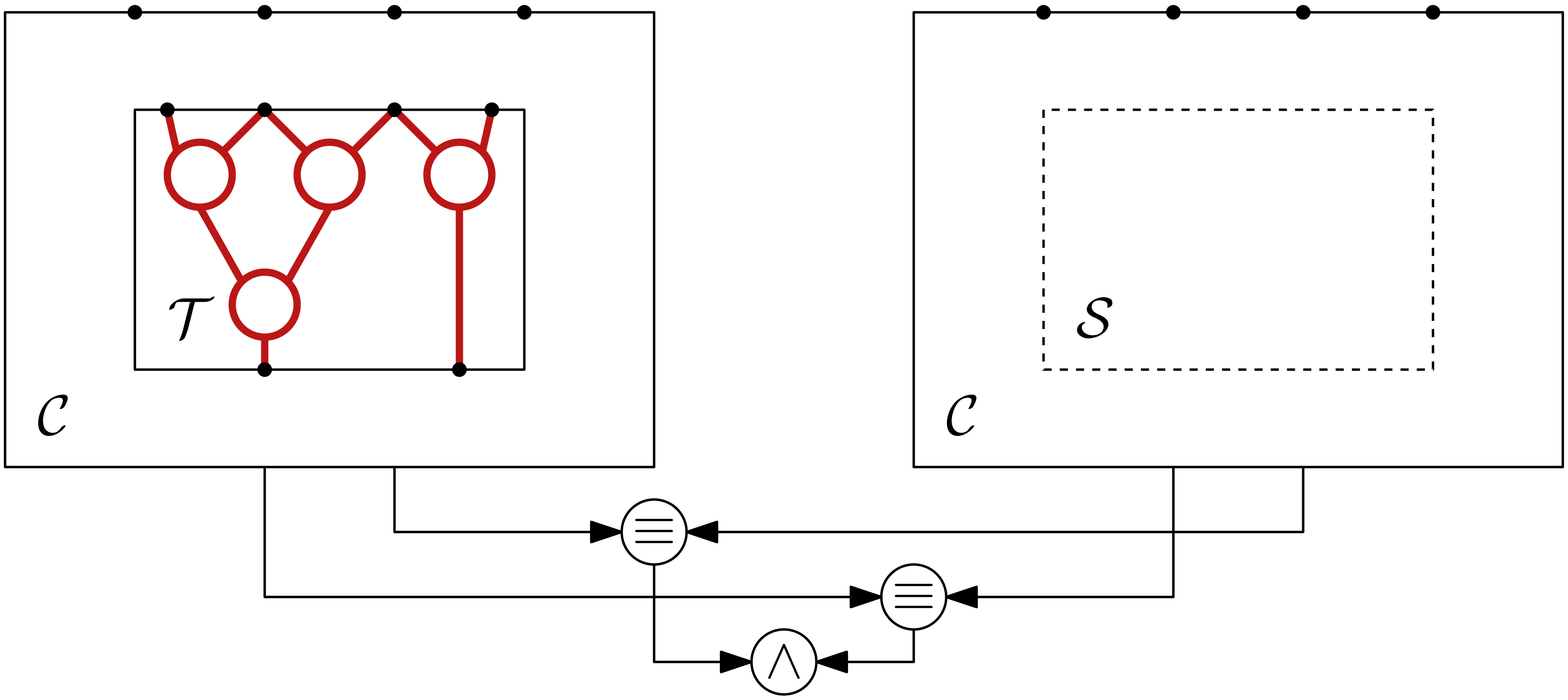
QBF-based Synthesis

$$\exists \mathcal{T}\, \forall I\, \exists G . \mathcal{C}[\mathcal{S} \leftarrow \mathcal{T}] \equiv \mathcal{C}$$



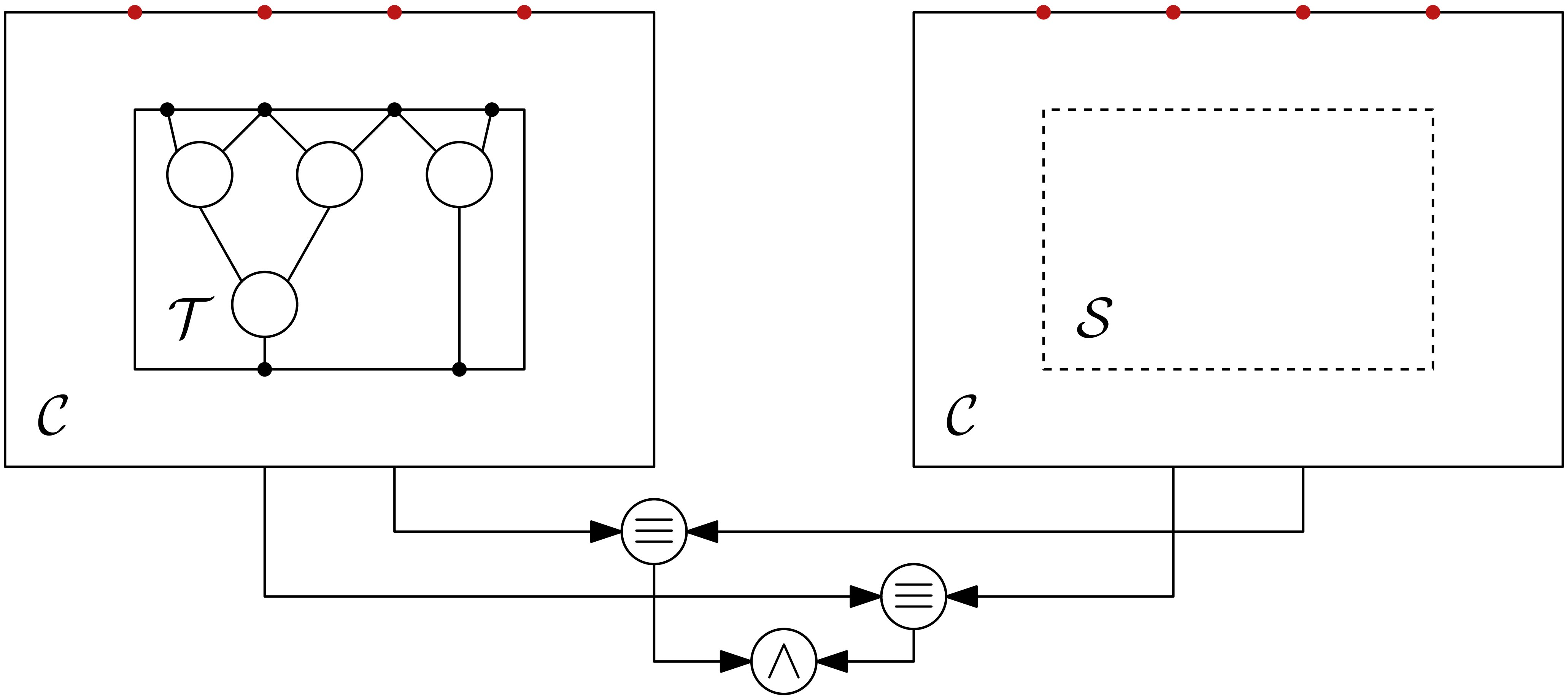
QBF-based Synthesis

$$\exists \mathcal{T}\, \forall I\, \exists G . \mathcal{C}[\mathcal{S} \leftarrow \mathcal{T}] \equiv \mathcal{C}$$



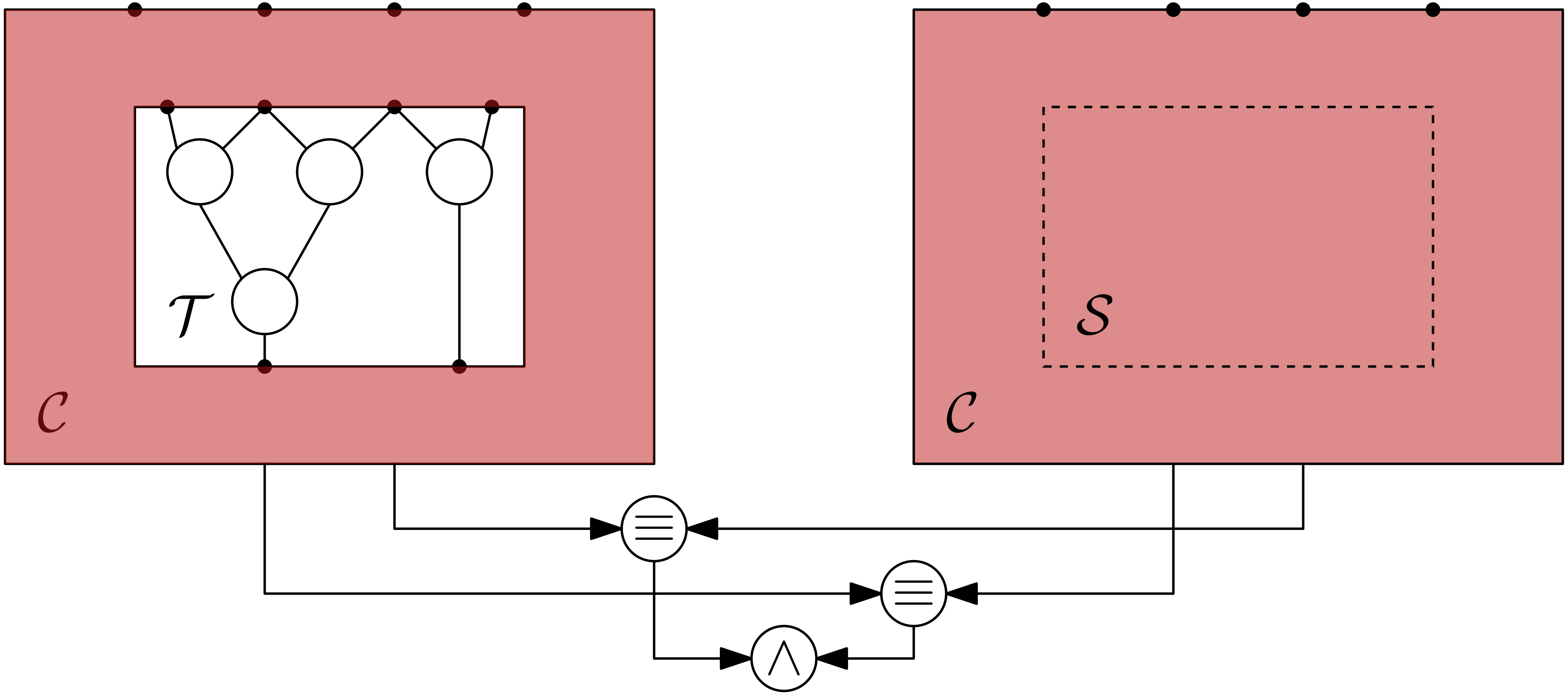
QBF-based Synthesis

$$\exists \mathcal{T}\, \forall I\, \exists G.\mathcal{C}[\mathcal{S} \leftarrow \mathcal{T}] \equiv \mathcal{C}$$



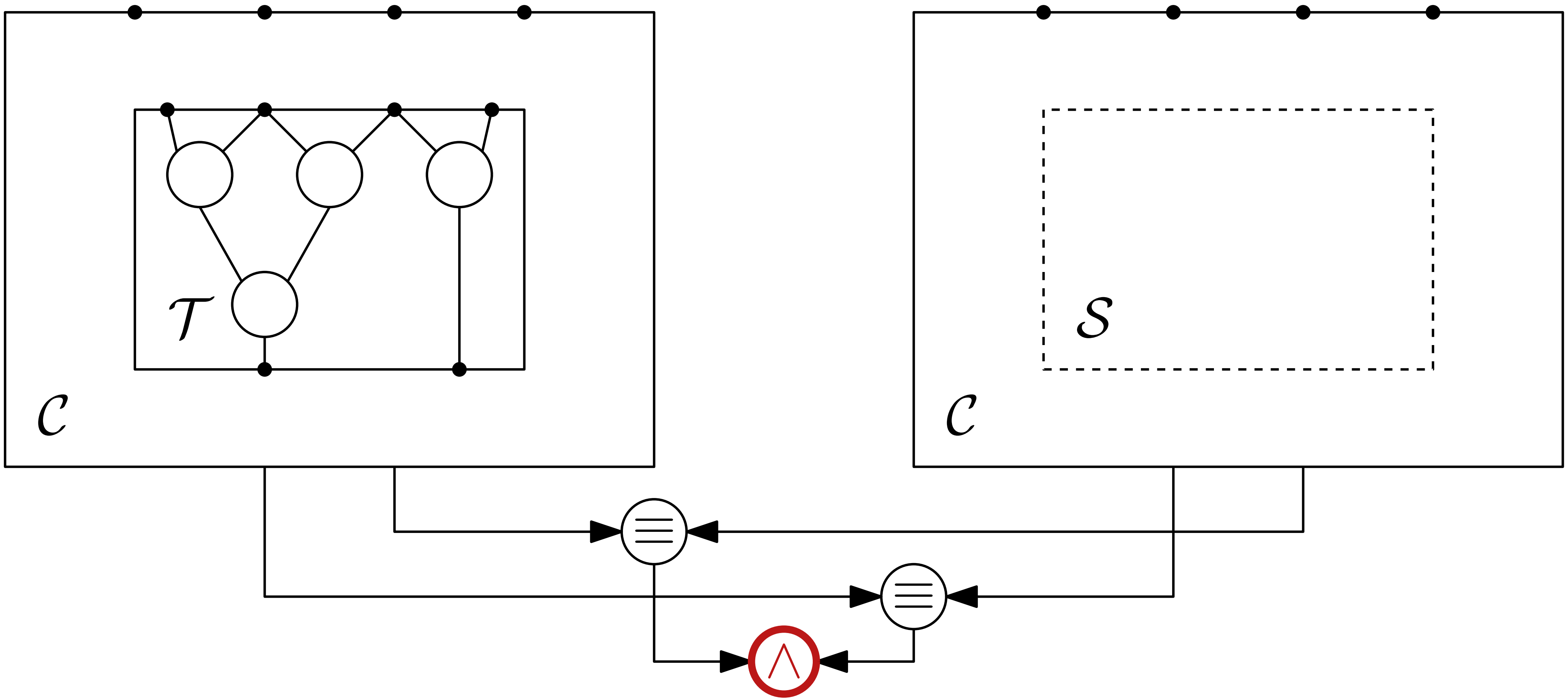
QBF-based Synthesis

$$\exists \mathcal{T}\, \forall I\, \exists G . \mathcal{C}[\mathcal{S} \leftarrow \mathcal{T}] \equiv \mathcal{C}$$



QBF-based Synthesis

$$\exists \mathcal{T}\, \forall I\, \exists G. \mathcal{C}[\mathcal{S} \leftarrow \mathcal{T}] \equiv \mathcal{C}$$



Boolean Relations

- Assign inputs to permitted outputs

Boolean Relations

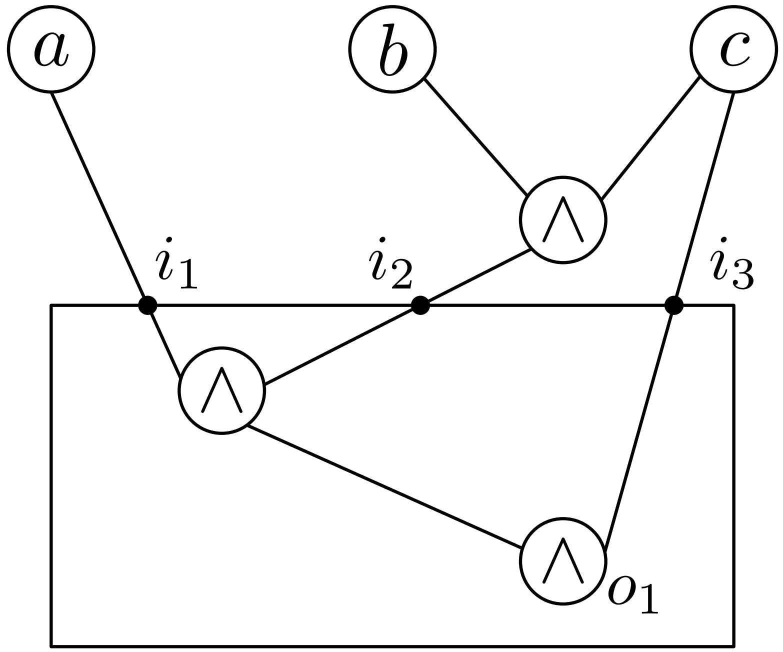
- Assign inputs to permitted outputs
- $R : \{0,1\}^{\mathrm{in}(\mathcal{S})} \rightarrow (\mathcal{P}(\{0,1\}^{\mathrm{out}(\mathcal{S})})\backslash\emptyset)$

Boolean Relations

- Assign inputs to permitted outputs
- $R : \{0,1\}^{\mathrm{in}(\mathcal{S})} \to (\mathcal{P}(\{0,1\}^{\mathrm{out}(\mathcal{S})}) \setminus \emptyset)$
- Allow to represent don't cares

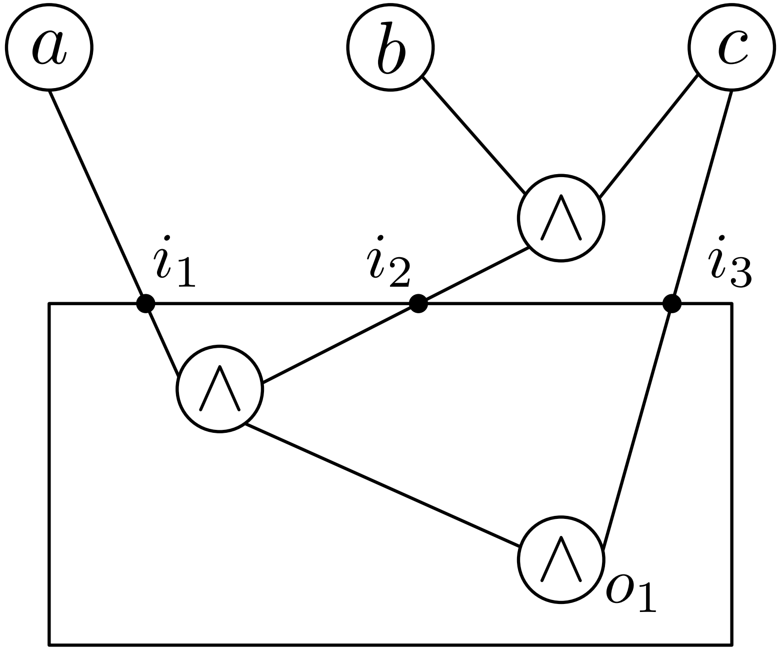
Boolean Relations

- Assign inputs to permitted outputs
- $R : \{0,1\}^{\mathrm{in}(\mathcal{S})} \to (\mathcal{P}(\{0,1\}^{\mathrm{out}(\mathcal{S})}) \setminus \emptyset)$
- Allow to represent don't cares



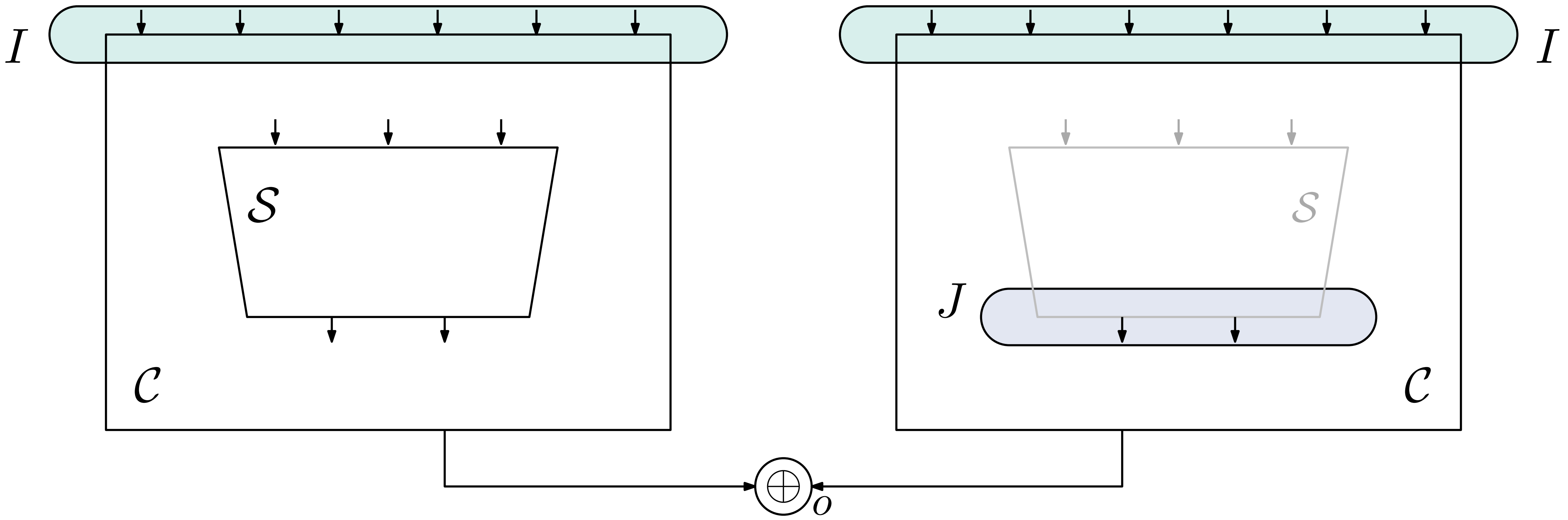
Boolean Relations

- Assign inputs to permitted outputs
- $R : \{0,1\}^{\mathrm{in}(\mathcal{S})} \to (\mathcal{P}(\{0,1\}^{\mathrm{out}(\mathcal{S})}) \setminus \emptyset)$
- Allow to represent don't cares



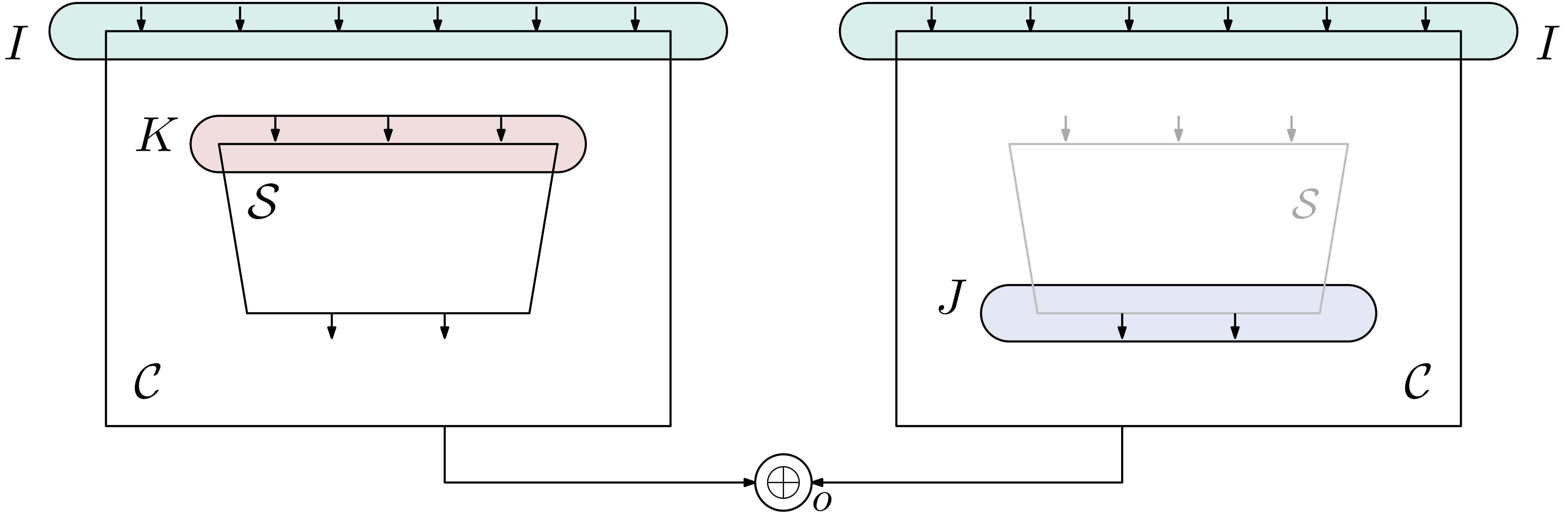
i_1	i_2	i_3	
0	0	0	$\{\neg o_1\}$
0	0	1	$\{\neg o_1\}$
0	1	0	$\{o_1, \neg o_1\}$
0	1	1	$\{\neg o_1\}$
1	0	0	$\{\neg o_1\}$
1	0	1	$\{\neg o_1\}$
1	1	0	$\{o_1, \neg o_1\}$
1	1	1	$\{o_1\}$

Computing Boolean Relations



Find assignments for I and J such that o is true

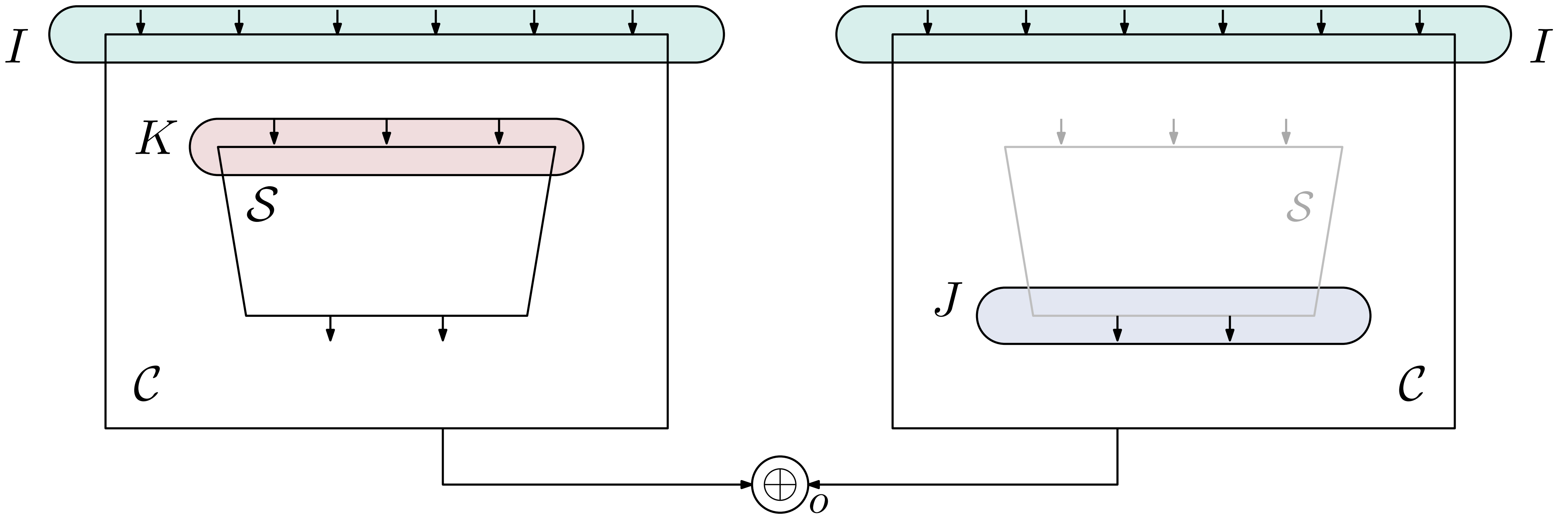
Computing Boolean Relations



Find assignments for I and J such that o is true

Under the assignment K, the subcircuit $\mathcal{S}$ must not attain J

Computing Boolean Relations



Find assignments for I and J such that o is true

Under the assignment K, the subcircuit $\mathcal{S}$ must not attain J

Iteratively compute assignments I, J, K by assumption based SAT solving

SAT-based Synthesis

Exact Synthesis

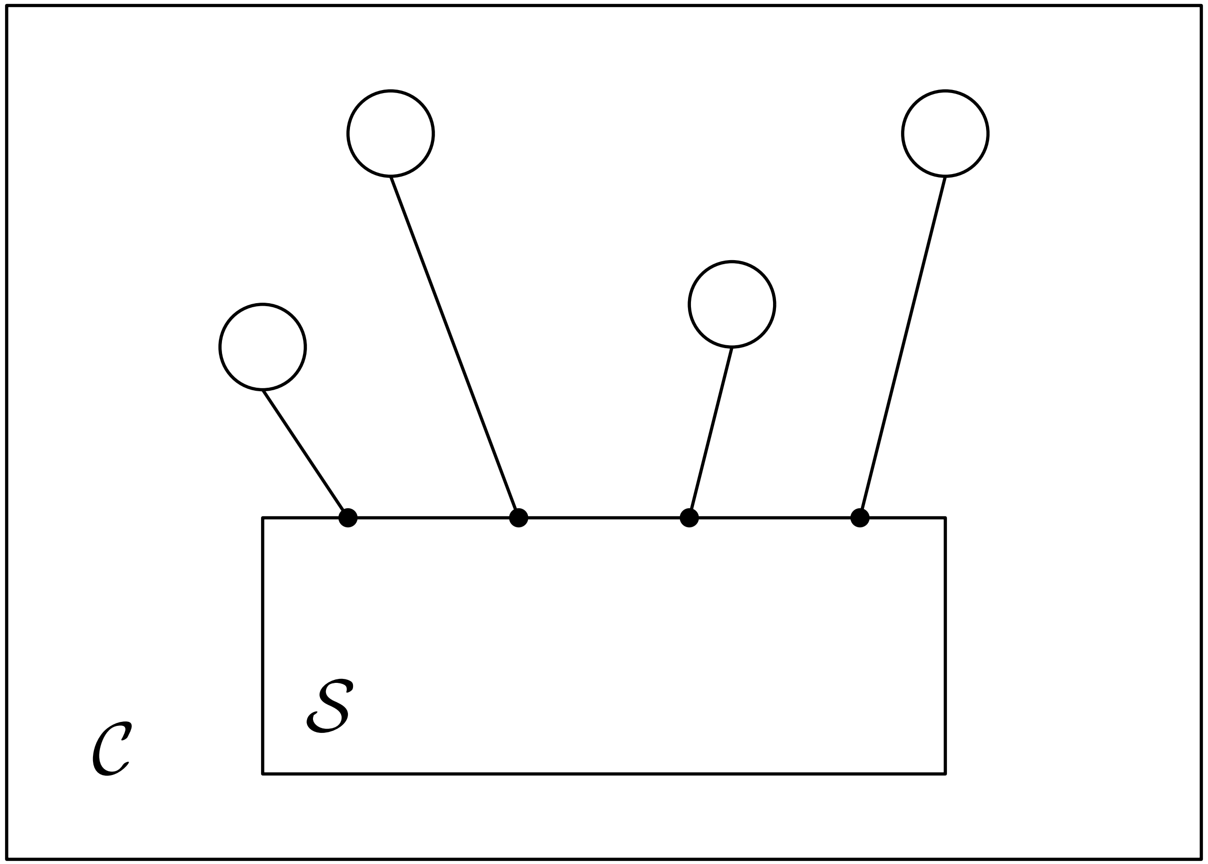
i_1	i_2	i_3	
0	0	0	$\neg o_1 \wedge \neg o_2$
0	0	1	$\neg o_1 \wedge o_2$
0	1	0	$\neg o_1 \wedge \neg o_2$
0	1	1	$\neg o_1 \wedge o_2$
1	0	0	$\neg o_1 \wedge \neg o_2$
1	0	1	$\neg o_1 \wedge o_2$
1	1	0	$o_1 \wedge o_2$
1	1	1	$o_1 \wedge o_2$

SAT-based Synthesis

Synthesising a relation

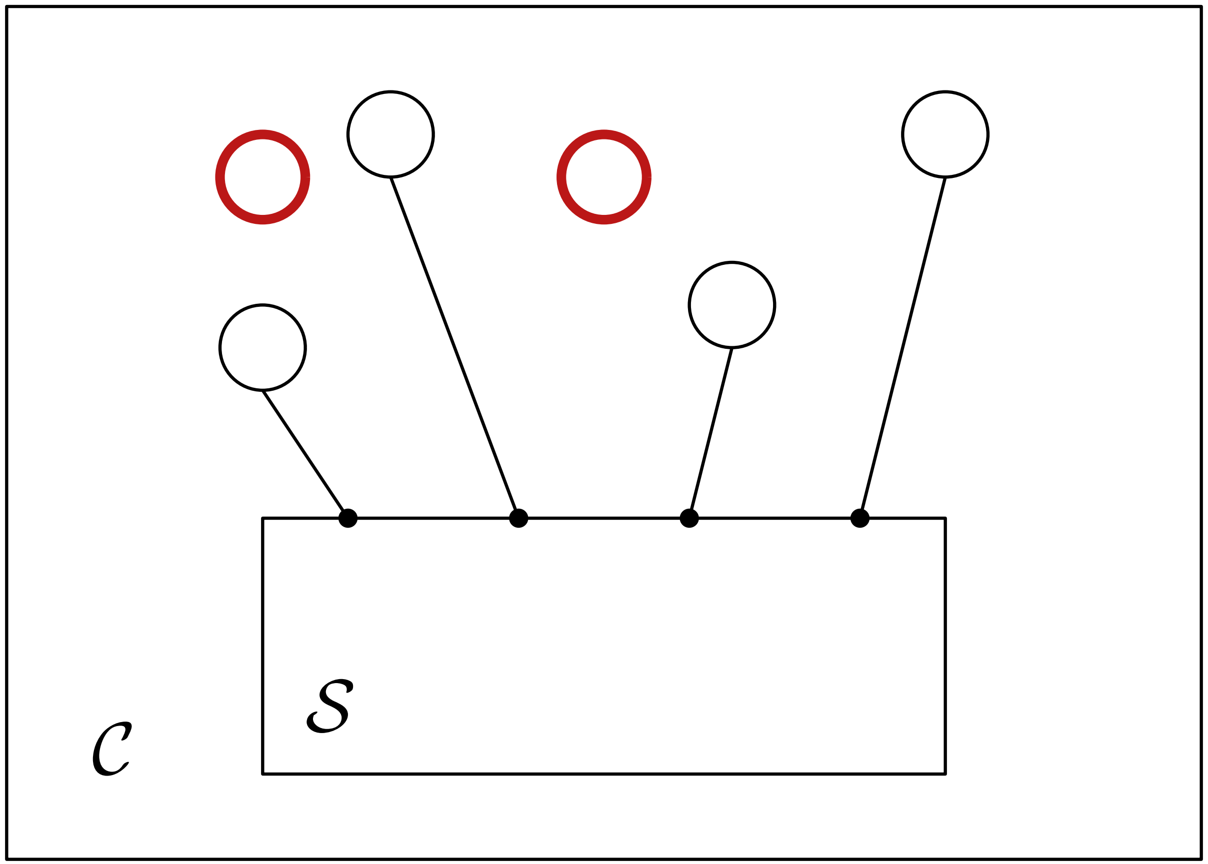
i_1	i_2	i_3	
0	0	0	$\neg o_1$
0	0	1	$(\neg o_1 \vee \neg o_2) \wedge (o_1 \vee o_2)$
0	1	0	$\neg o_1 \vee \neg o_2$
0	1	1	o_2
1	0	0	$(\neg o_1 \vee o_2) \wedge (o_1 \vee \neg o_2)$
1	0	1	$(\neg o_1 \vee \neg o_2) \wedge (o_1 \vee o_2)$
1	1	0	$\neg o_1 \vee \neg o_2$
1	1	1	o_1

Additional Inputs



$\mathcal{C}$ can contain nodes with functionality used in $\mathcal{S}$

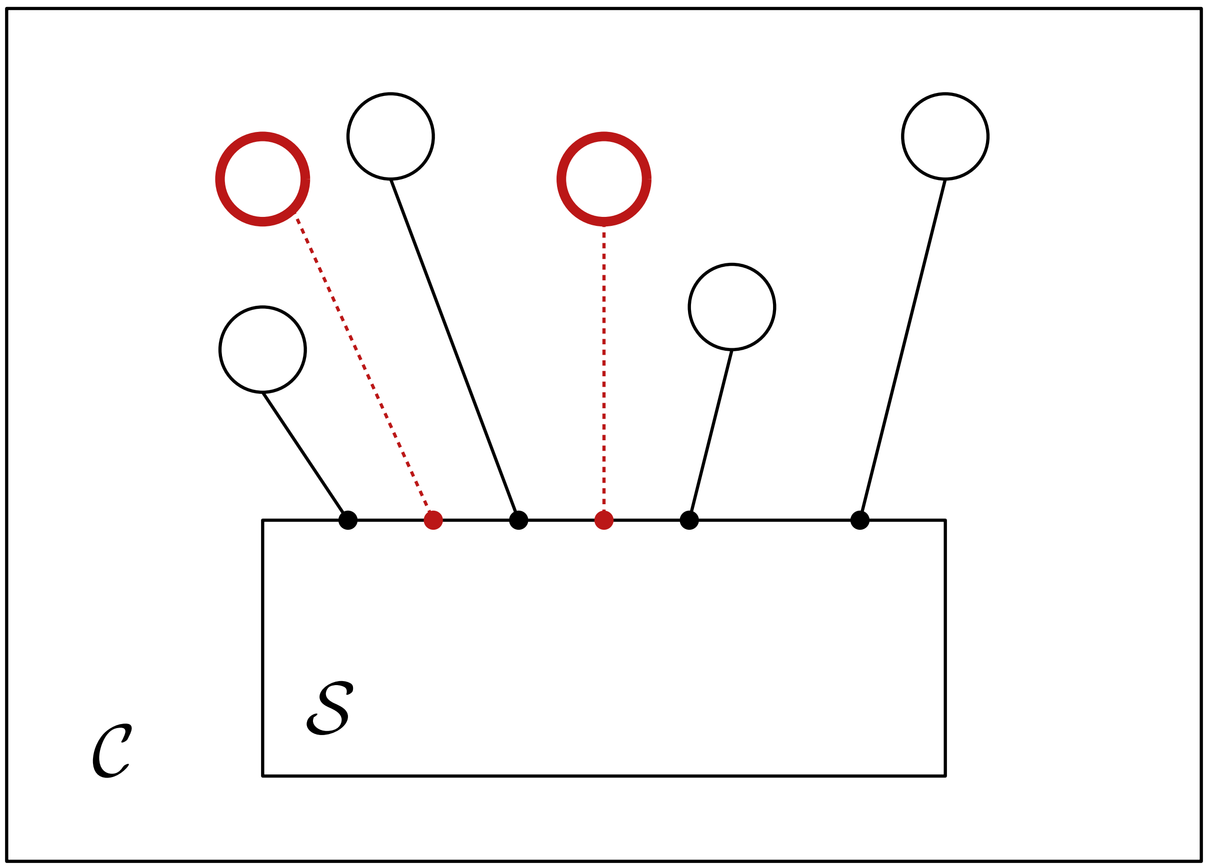
Additional Inputs



$\mathcal{C}$ can contain nodes with functionality used in $\mathcal{S}$

Select additional nodes

Additional Inputs

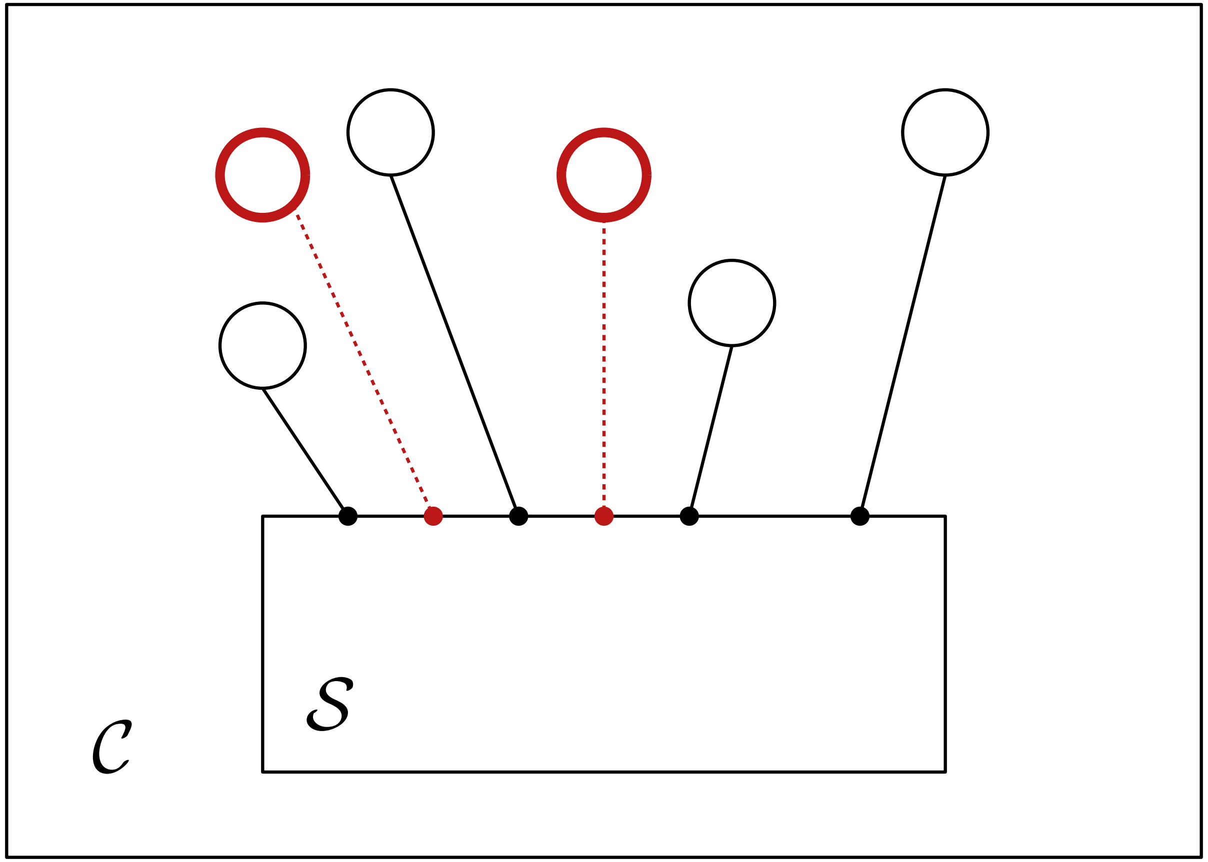


$\mathcal{C}$ can contain nodes with functionality used in $\mathcal{S}$

Select additional nodes

Allow these nodes as inputs when synthesing replacement

Additional Inputs



$\mathcal{C}$ can contain nodes with functionality used in $\mathcal{S}$

Select additional nodes

Allow these nodes as inputs when synthesing replacement

Works better with QBF-based synthesis

Availability

- Implemented in our tool eSLIM

  `https://github.com/fxreichl/eSLIM`

Availability

- Implemented in our tool eSLIM

  `https://github.com/fxreichl/eSLIM`

- Integrated within ABC

  Available with the `&eslim` command

  Currently limited to SAT-based approach