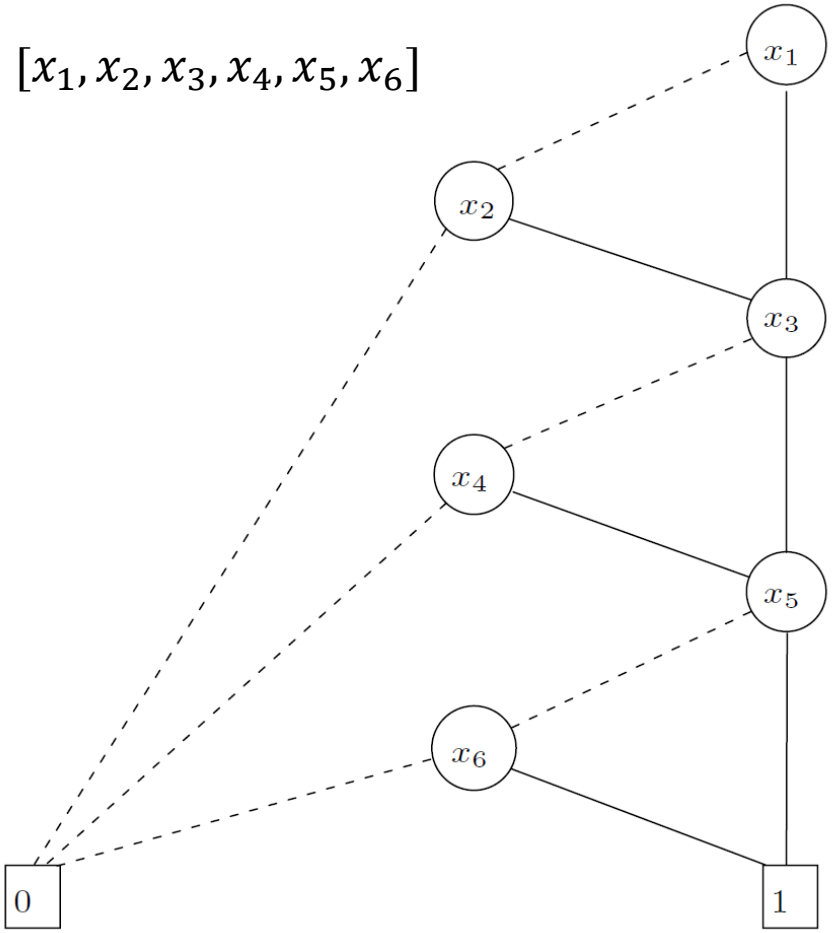
Übung 10 – BDDs & symbolic model checking
Formale Techniken in der Software-Entwicklung

Christian Kroiß

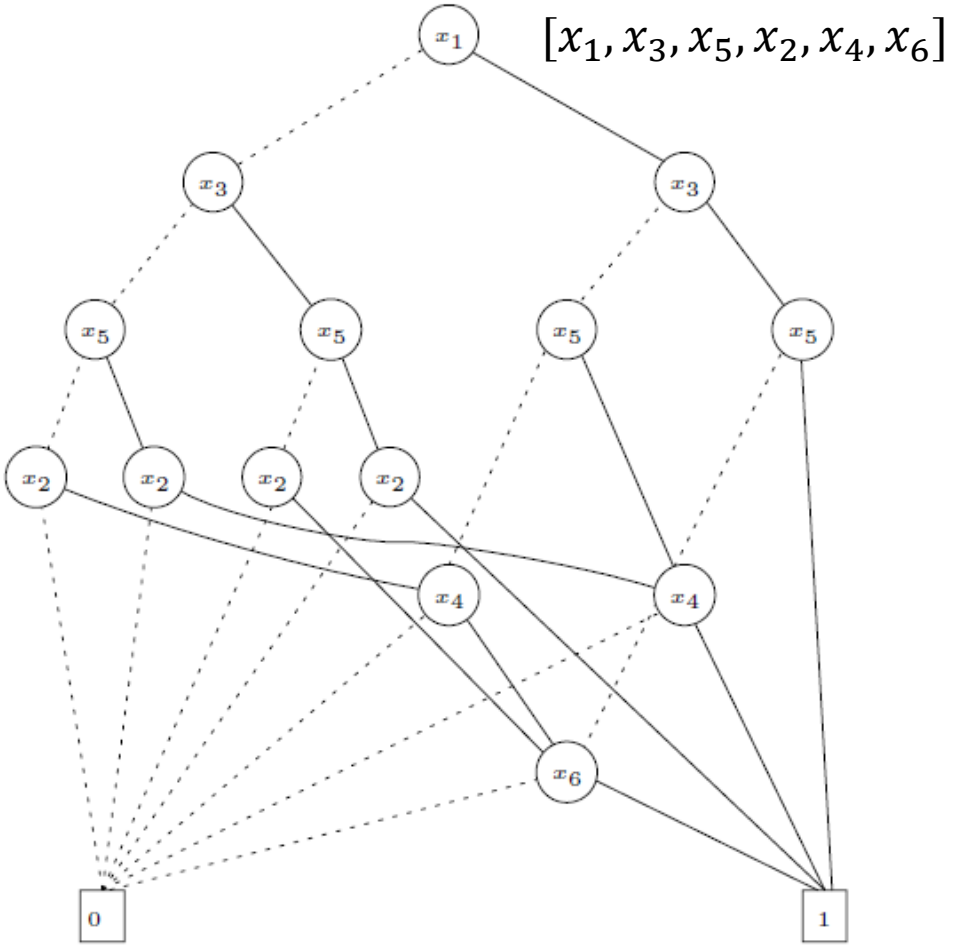
Impact of variable ordering

$$(x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (x_5 \vee x_6)$$

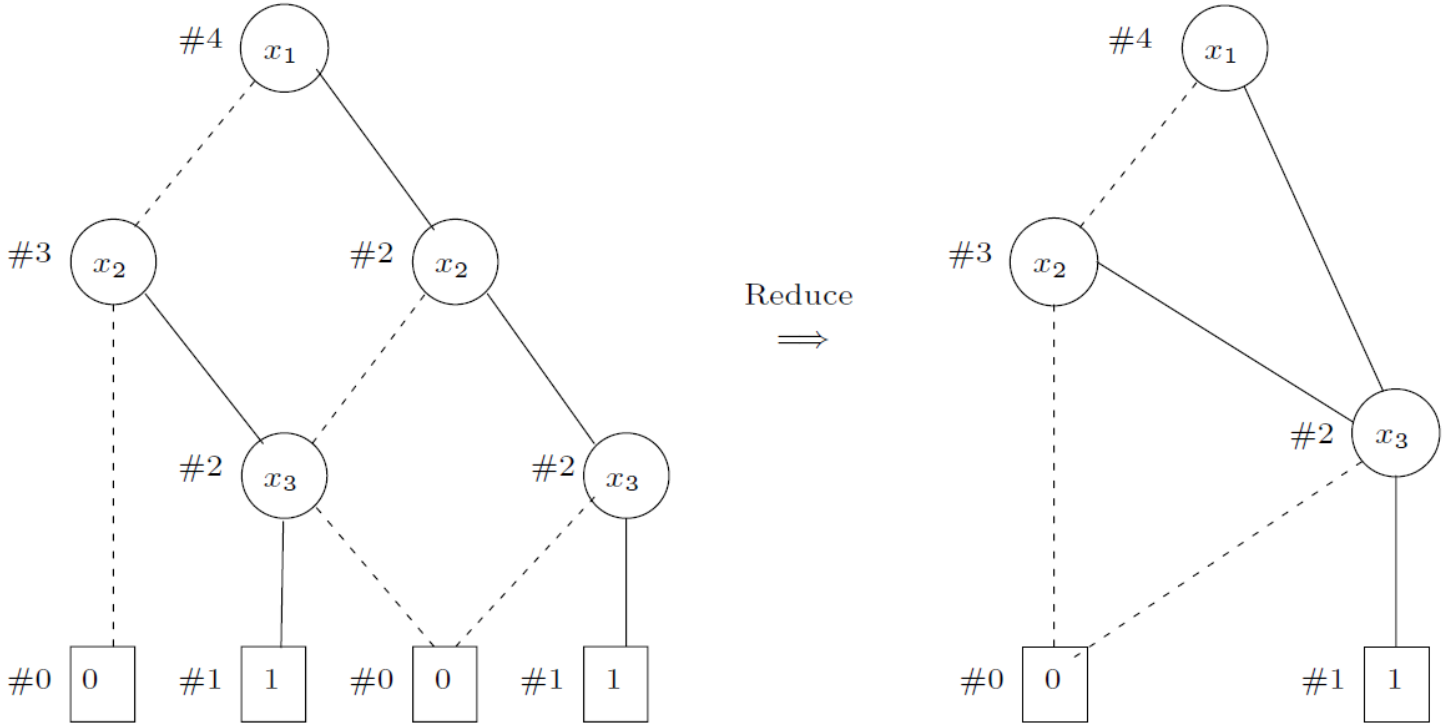
$[x_1, x_2, x_3, x_4, x_5, x_6]$



$[x_1, x_3, x_5, x_2, x_4, x_6]$

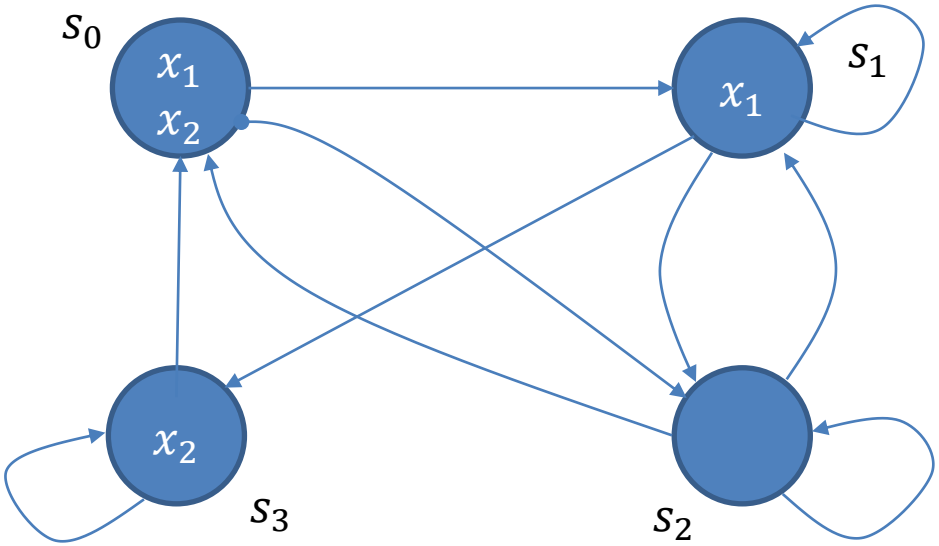


Algorithm reduce



- If the label id(lo(n)) is the same as id(hi(n)), then we set id(n) to be that label. In other words, node n performs a redundant test and can be eliminated by reduction C2.
- If there is another node m such that n and m have the same variable xi, and id(lo(n)) = id(lo(m)) and id(hi(n)) = id(hi(m)), then we set id(n) to be id(m). This is because the nodes n and m compute the same boolean function (compare with reduction C3).
- Otherwise, we set id(n) to the next unused integer label.

Aufgabe 1



a) Work out the truth table for the transition relation, ordering the columns $[x_1, x_1', x_2, x_2']$. There should be as many 1s in the final column as there are arrows in the transition relation.

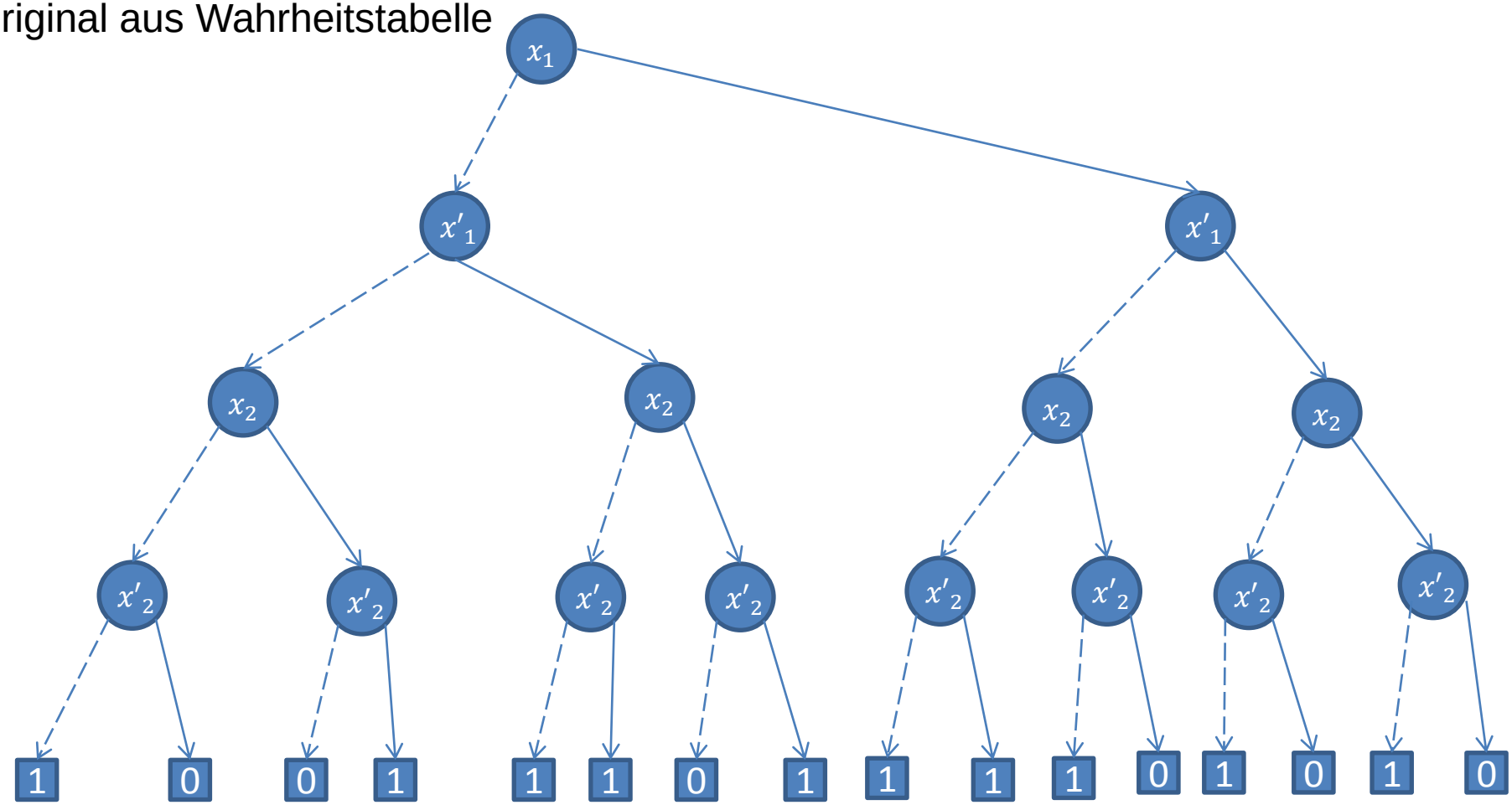
b) Draw the OBDD (ordered BDD) for this transition relation using the variable ordering $[x_1, x_1', x_2, x_2']$.

Aufgabe 1-a)

X1	X1'	X2	X2'	→	
0	0	0	0	1	s2→s2
0	0	0	1	0	s2→s3
0	0	1	0	0	s3→s2
0	0	1	1	1	s3→s3
0	1	0	0	1	s2→s1
0	1	0	1	1	s2→ s0
0	1	1	0	0	s3→s1
0	1	1	1	1	s3->s0
1	0	0	0	1	s1→s2
1	0	0	1	1	s1→s3
1	0	1	0	1	s0→s2
1	0	1	1	0	s0→s3
1	1	0	0	1	s1→s1
1	1	0	1	0	s1→s0
1	1	1	0	1	s0→s1
1	1	1	1	0	s0→s0

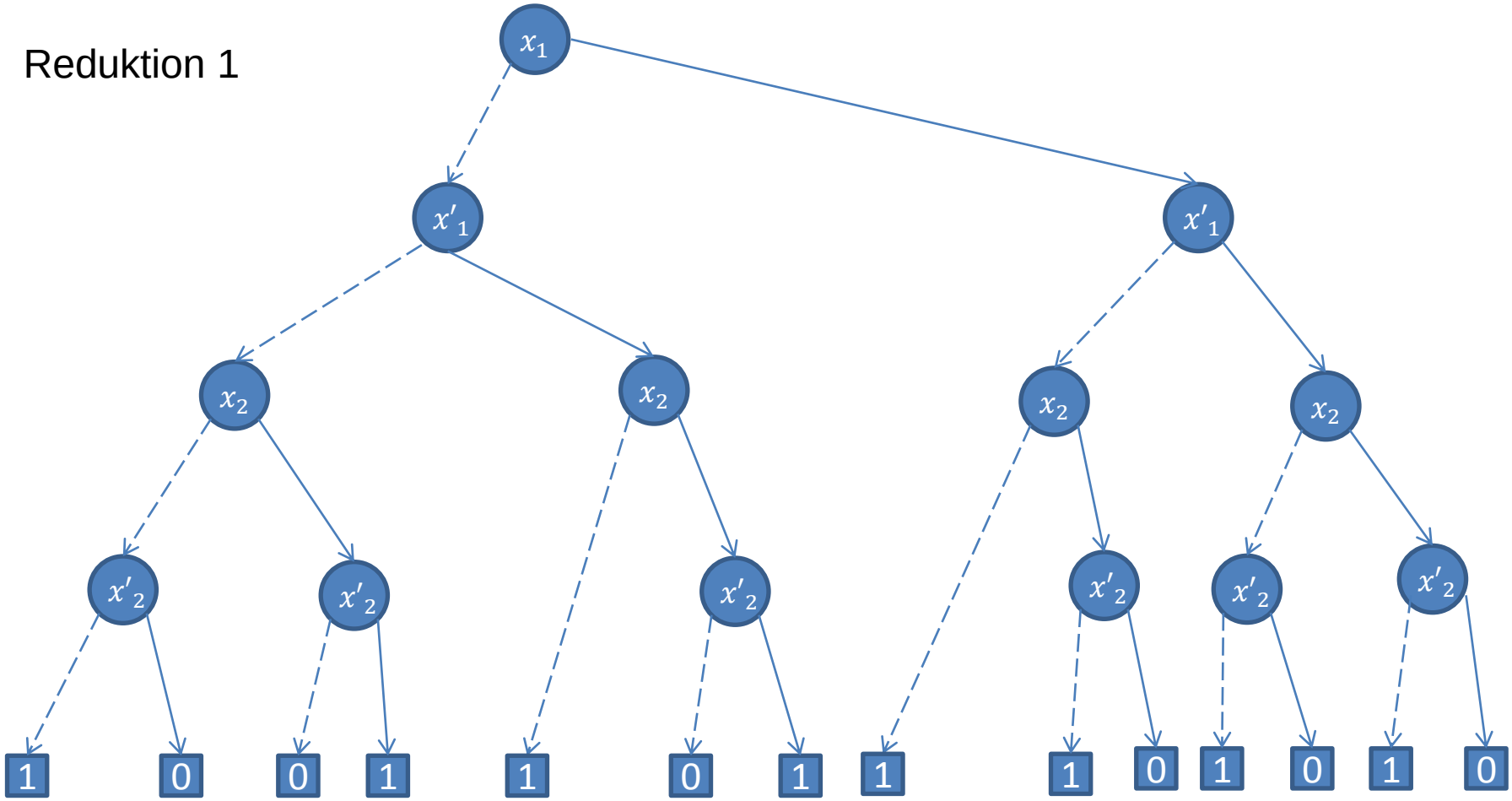
Aufgabe 1-b)

Original aus Wahrheitstabelle



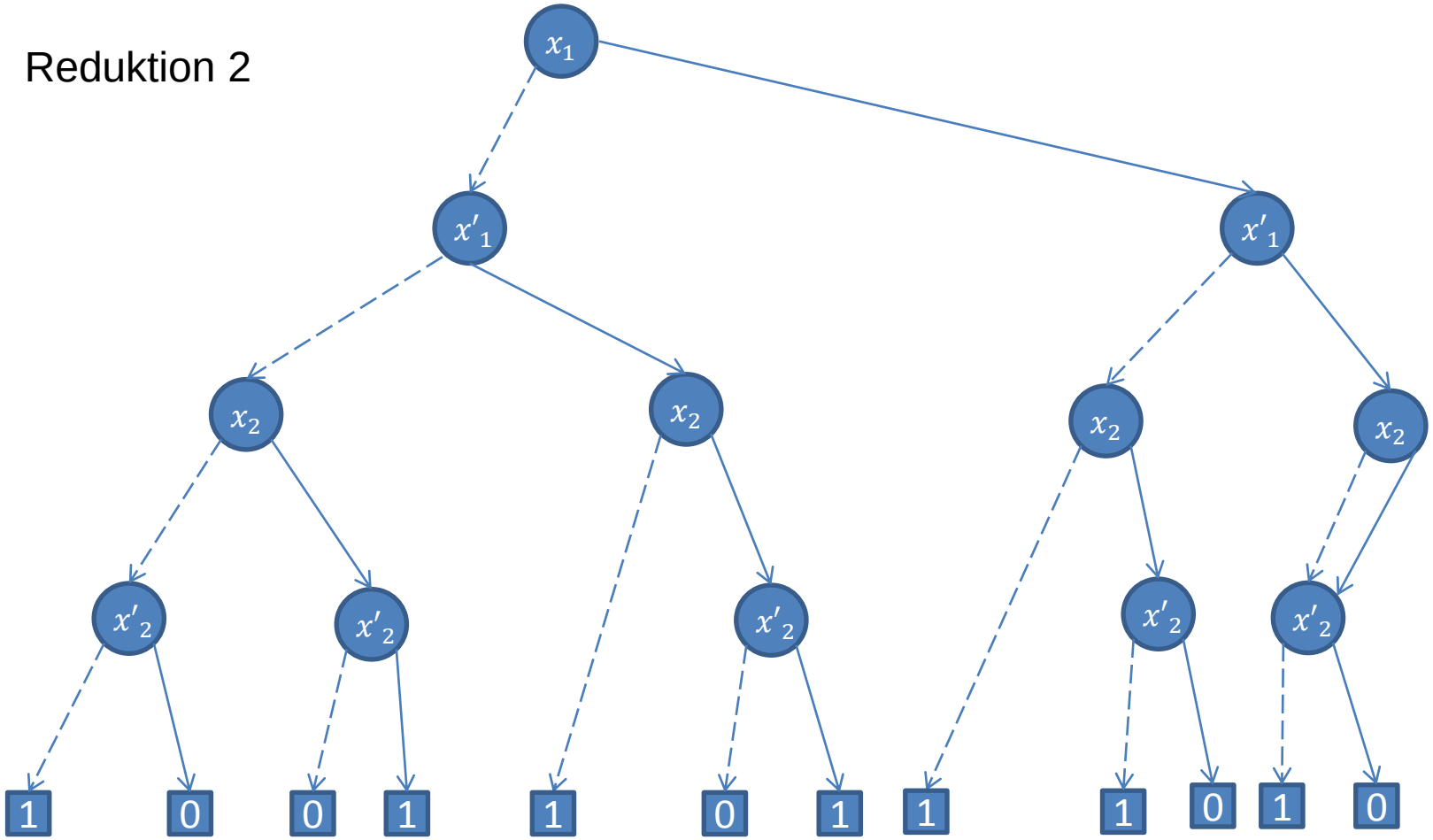
Aufgabe 1-b)

Reduktion 1



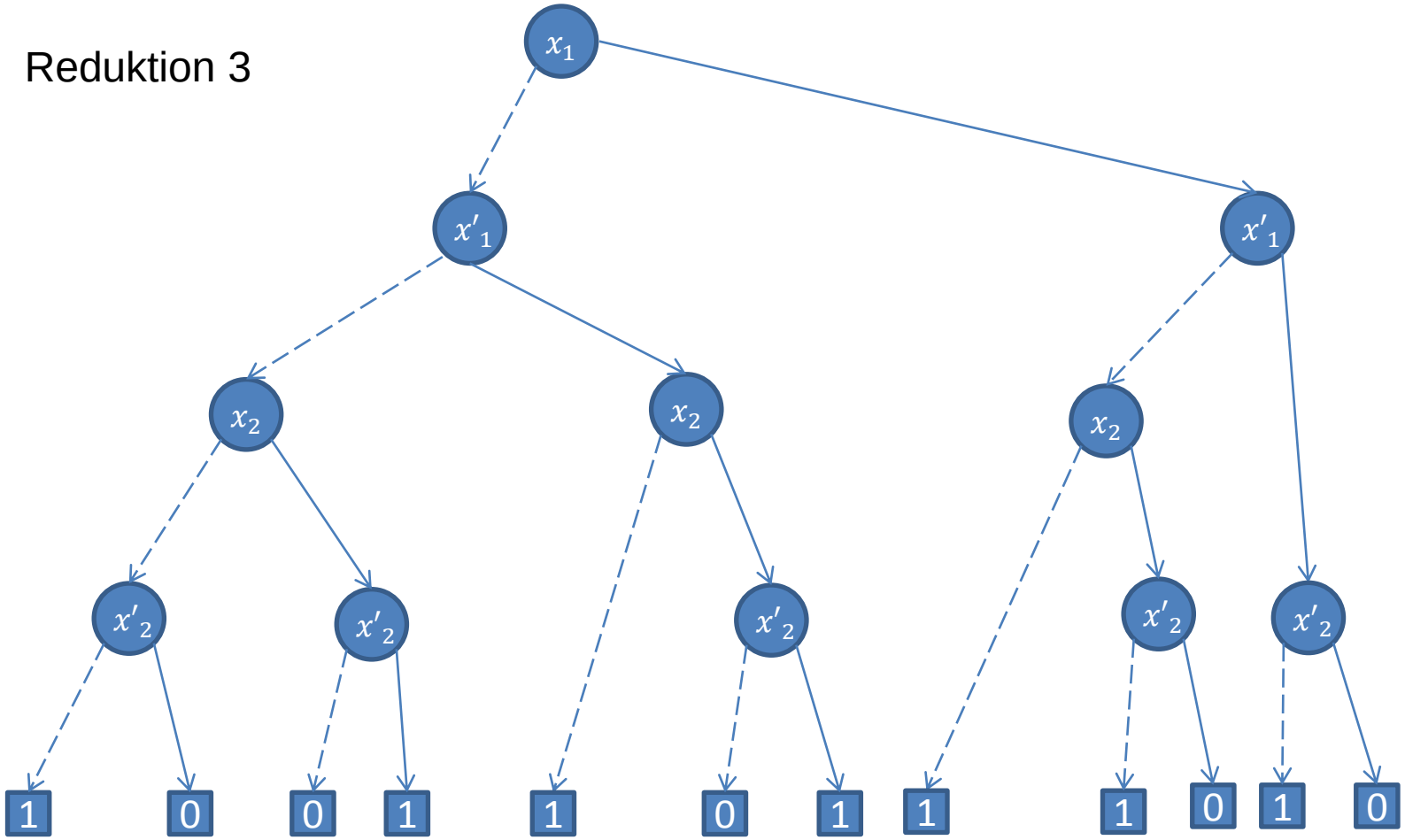
Aufgabe 1-b)

Reduktion 2

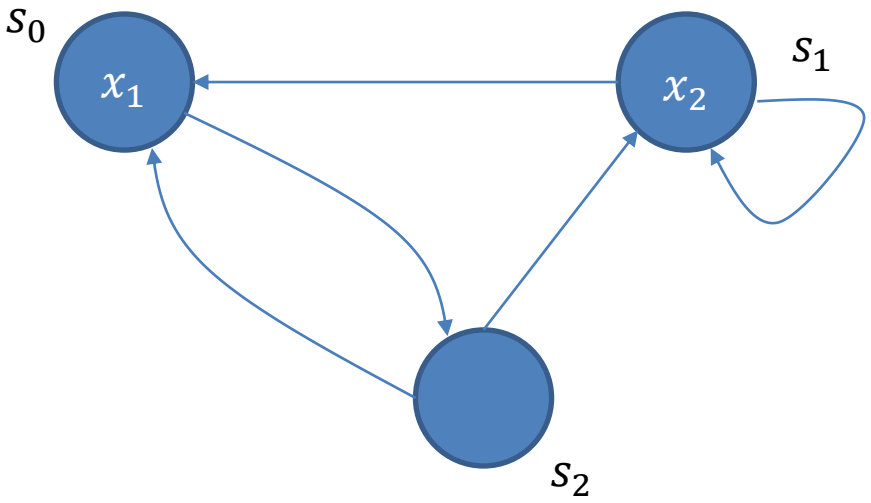


Aufgabe 1-b)

Reduktion 3



Aufgabe 2



Apply the CTL model checking algorithm, but now interpreted over OBDDs in the ordering $[x_1, x_2]$, to compute the set of states of the model above which satisfy

$$AG\ (x_1 \vee \neg x_2)$$

Show the BDDs which are computed along the way.

Aufgabe 2

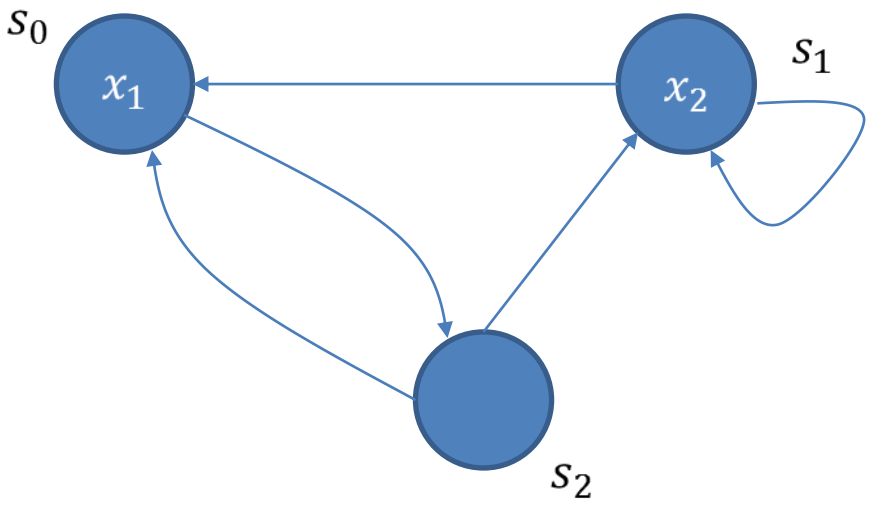
Umformung:

$$AG(x_1 \vee \neg x_2) = \neg EF \neg (x_1 \vee \neg x_2)$$
$$= \neg (true\, EU\, (\neg x_1 \wedge x_2))$$

$$\varphi := \neg x_1 \wedge x_2$$

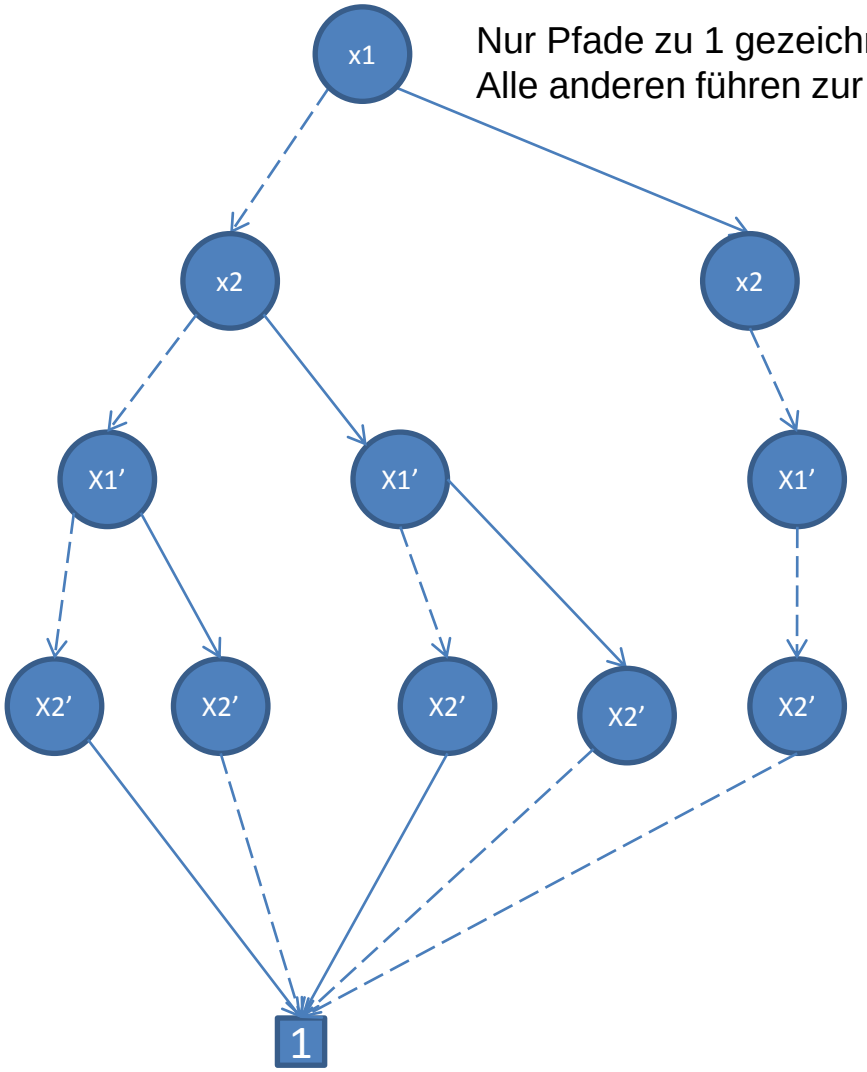
$$[\![true EU \varphi]\!] \text{ kleinster Fixpunkt von } \pi : Z \mapsto [\![\varphi]\!] \cup R^{-1}(Z)$$

Aufgabe 2



X1	X2	X1'	X2'	→	
0	0	0	0	0	s2→s2
0	0	0	1	1	s2→s1
0	1	0	0	0	s1→s2
0	1	0	1	1	s1→s1
0	0	1	0	1	s2→s0
0	0	1	1	0	-
0	1	1	0	1	s2→s0
0	1	1	1	0	-
1	0	0	0	1	s0→s2
1	0	0	1	0	s0→s1
1	1	0	0	0	-
1	1	0	1	0	-
1	0	1	0	0	s0→s0
1	0	1	1	0	-
1	1	1	0	0	-
1	1	1	1	0	-

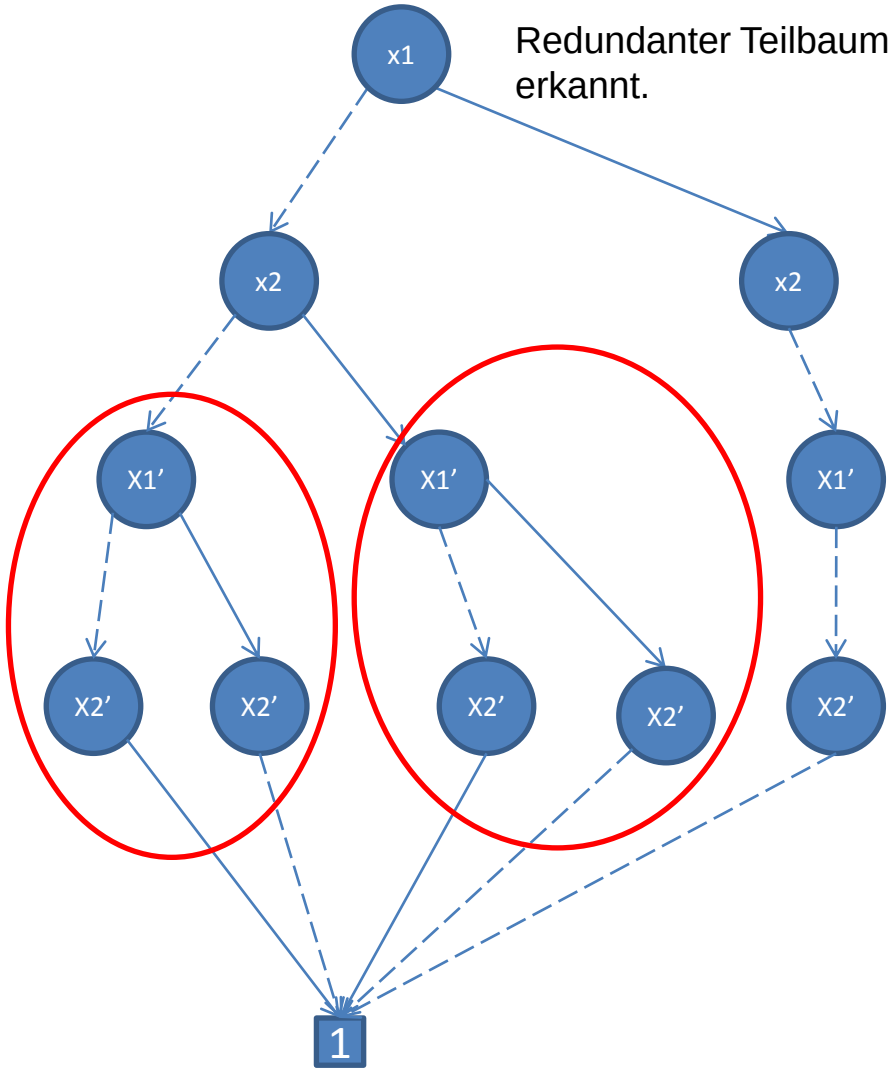
Aufgabe 2



Nur Pfade zu 1 gezeichnet.
Alle anderen führen zur 0.

X1	X2	X1'	X2'	→	
0	0	0	0	0	s2→s2
0	0	0	1	1	s2→s1
0	1	0	0	0	s1→s2
0	1	0	1	1	s1→s1
0	0	1	0	1	s2→s0
0	0	1	1	0	-
0	1	1	0	1	s2→s0
0	1	1	1	0	-
1	0	0	0	1	s0→s2
1	0	0	1	0	s0→s1
1	1	0	0	0	-
1	1	0	1	0	-
1	0	1	0	0	s0→s0
1	0	1	1	0	-
1	1	1	0	0	-
1	1	1	1	0	-

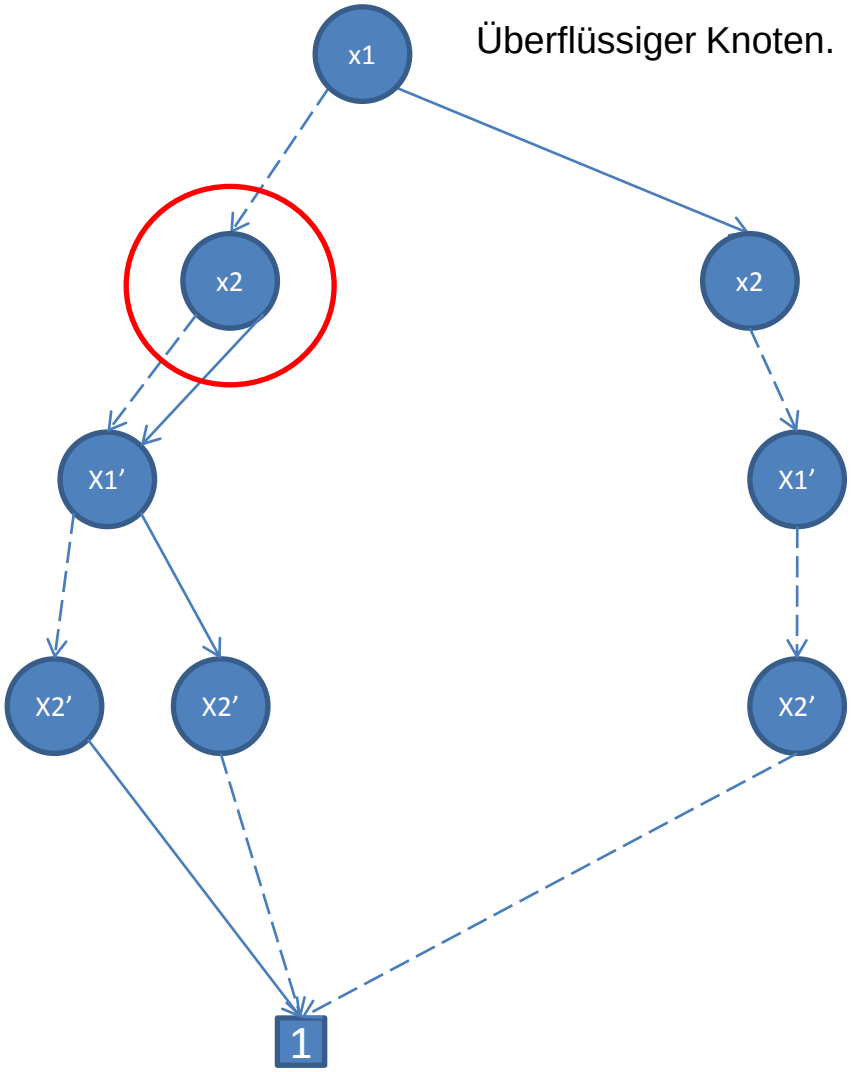
Aufgabe 2



Redundanter Teilbaum erkannt.

X1	X2	X1'	X2'	→	
0	0	0	0	0	s2→s2
0	0	0	1	1	s2→s1
0	1	0	0	0	s1→s2
0	1	0	1	1	s1→s1
0	0	1	0	1	s2→s0
0	0	1	1	0	-
0	1	1	0	1	s2→s0
0	1	1	1	0	-
1	0	0	0	1	s0→s2
1	0	0	1	0	s0→s1
1	1	0	0	0	-
1	1	0	1	0	-
1	0	1	0	0	s0→s0
1	0	1	1	0	-
1	1	1	0	0	-
1	1	1	1	0	-

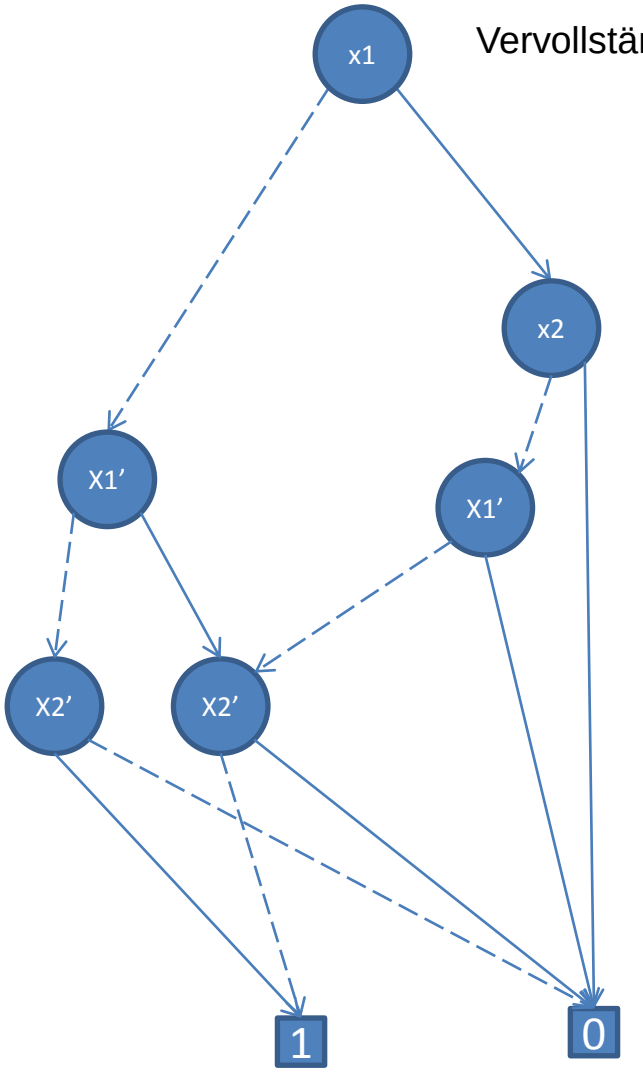
Aufgabe 2



Überflüssiger Knoten.

X1	X2	X1'	X2'	→	
0	0	0	0	0	s2→s2
0	0	0	1	1	s2→s1
0	1	0	0	0	s1→s2
0	1	0	1	1	s1→s1
0	0	1	0	1	s2→s0
0	0	1	1	0	-
0	1	1	0	1	s2→s0
0	1	1	1	0	-
1	0	0	0	1	s0→s2
1	0	0	1	0	s0→s1
1	1	0	0	0	-
1	1	0	1	0	-
1	0	1	0	0	s0→s0
1	0	1	1	0	-
1	1	1	0	0	-
1	1	1	1	0	-

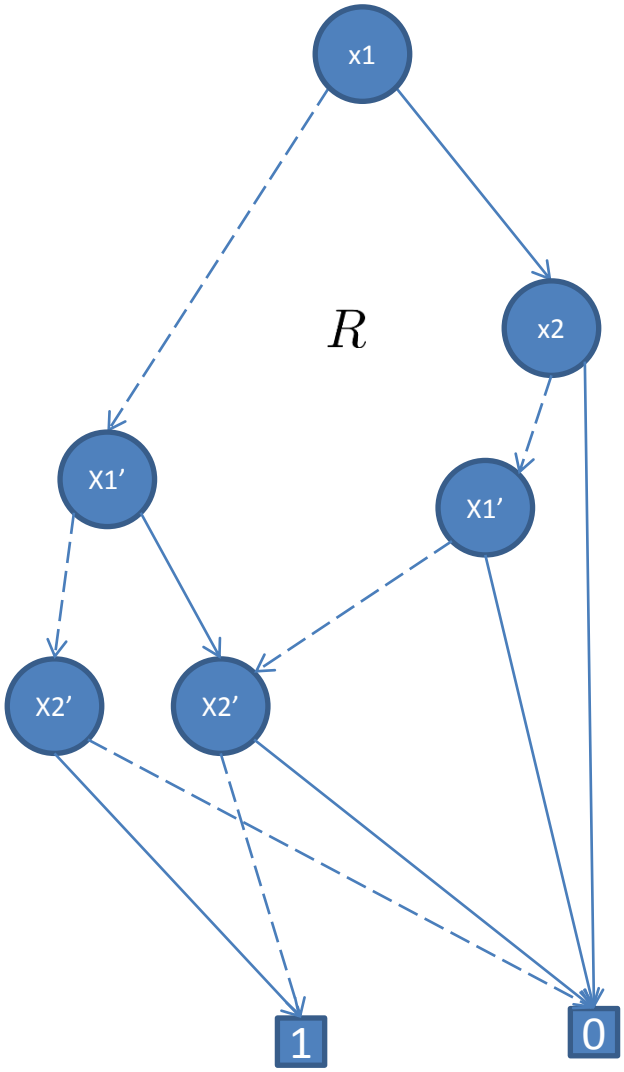
Aufgabe 2



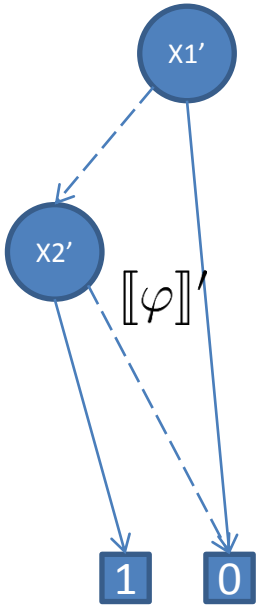
Vervollständigen

X1	X2	X1'	X2'	→	
0	0	0	0	0	s2→s2
0	0	0	1	1	s2→s1
0	1	0	0	0	s1→s2
0	1	0	1	1	s1→s1
0	0	1	0	1	s2→s0
0	0	1	1	0	-
0	1	1	0	1	s2→s0
0	1	1	1	0	-
1	0	0	0	1	s0→s2
1	0	0	1	0	s0→s1
1	1	0	0	0	-
1	1	0	1	0	-
1	0	1	0	0	s0→s0
1	0	1	1	0	-
1	1	1	0	0	-
1	1	1	1	0	-

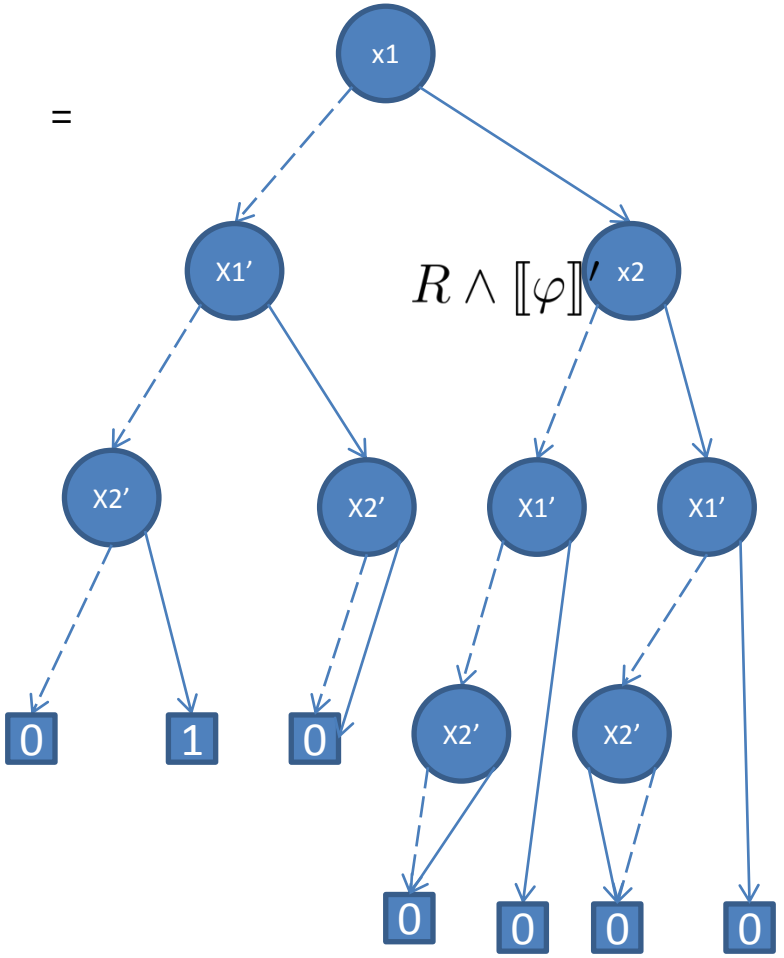
Aufgabe 2



AND

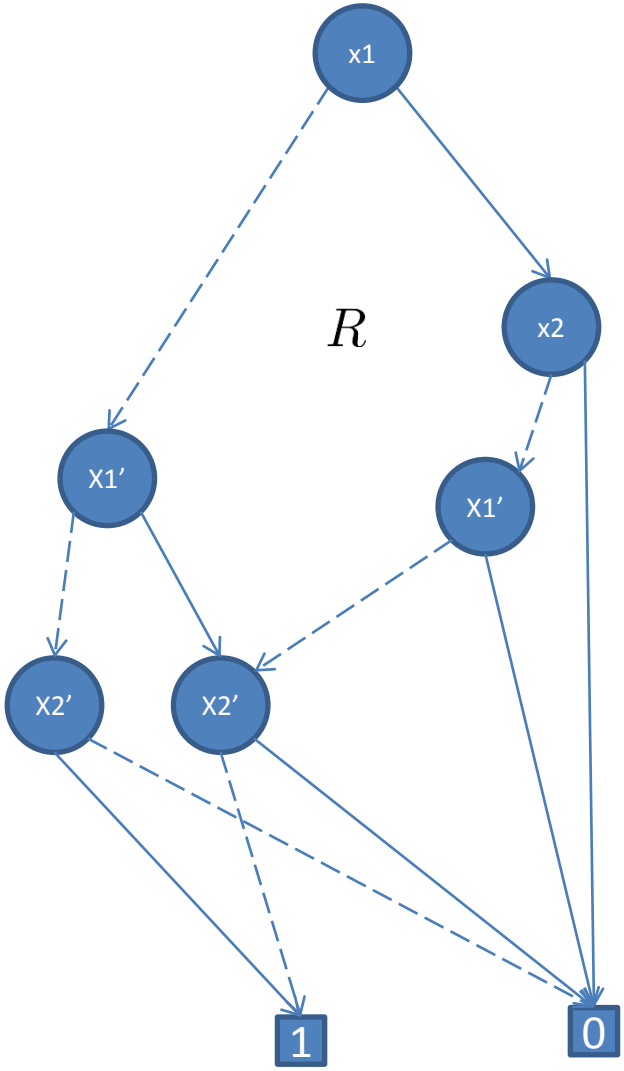


=



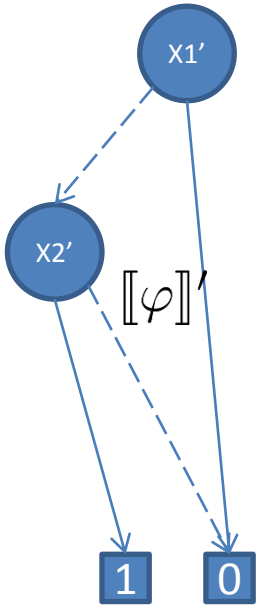
$$\varphi = x_1 \vee \neg x_2$$

Aufgabe 2



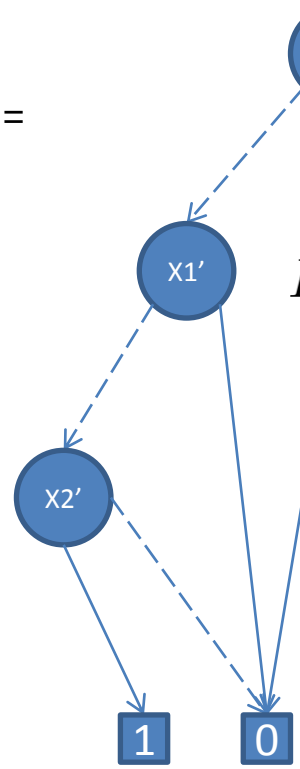
R

AND



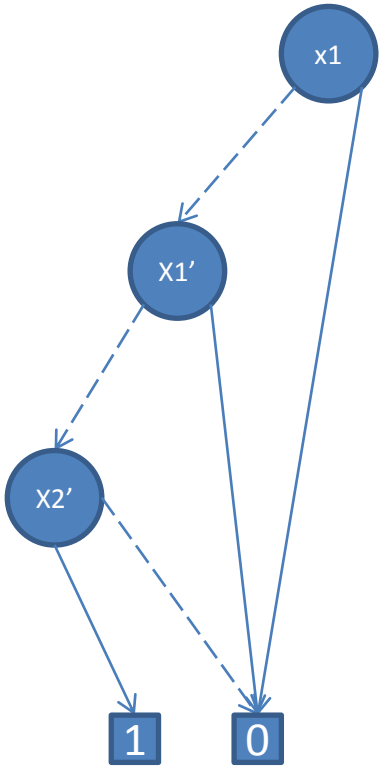
$[\![\varphi]\!]'$

=

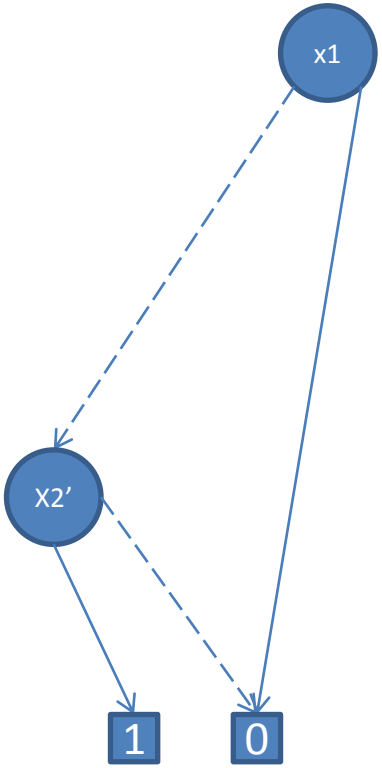


$R \wedge [\![\varphi]\!]' =: B^1_{AND}$

Aufgabe 2



$B^1_{AND}|_{x'_1=false}$

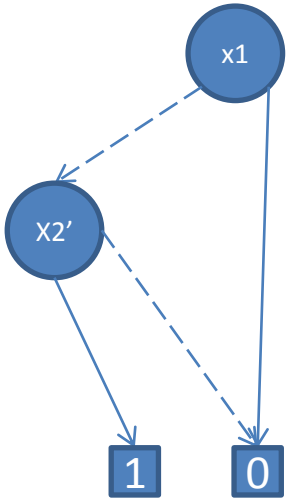


$B^1_{AND}|_{x'_1=false}$

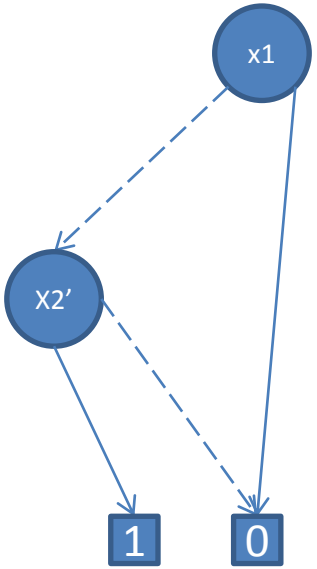
0

OR

$\exists x'_1 B^1_{AND} =: B^1_{\exists x'_1}$

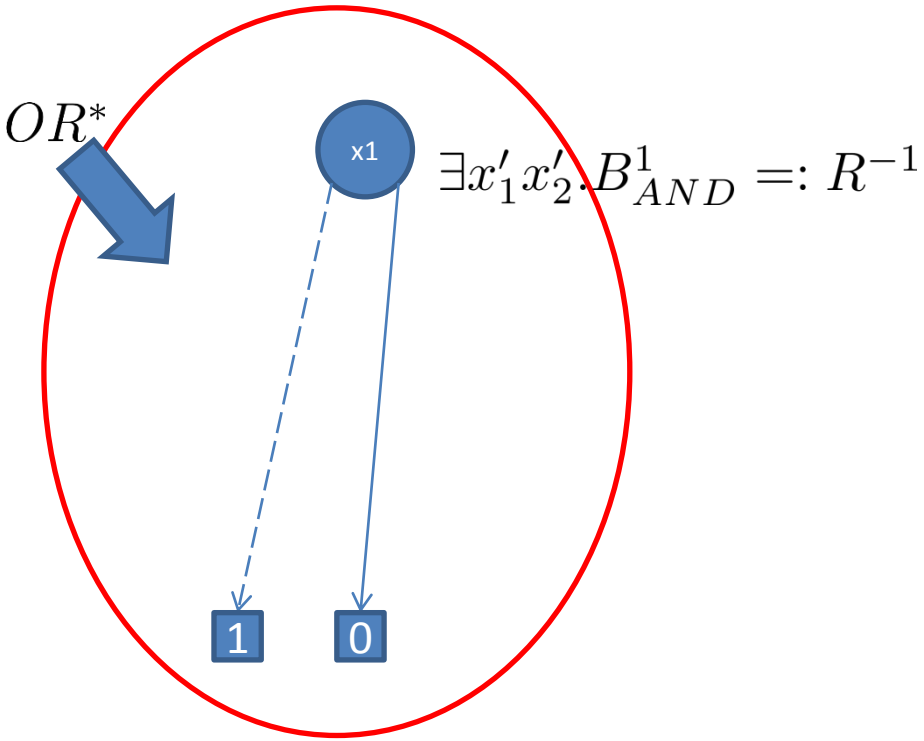


$B^1_{\exists x'_1}$



$B^1_{\exists x'_1}|_{x'_2=false}$

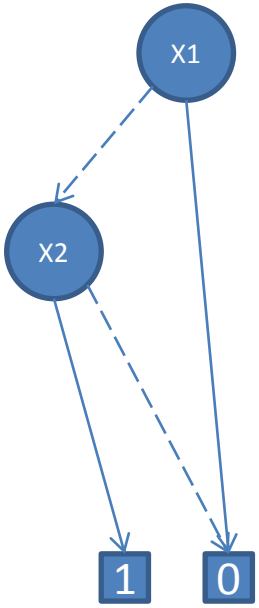
$B^1_{\exists x'_1}|_{x'_2=true}$

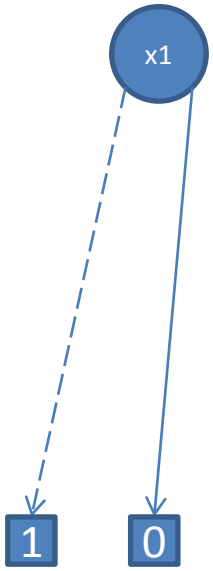


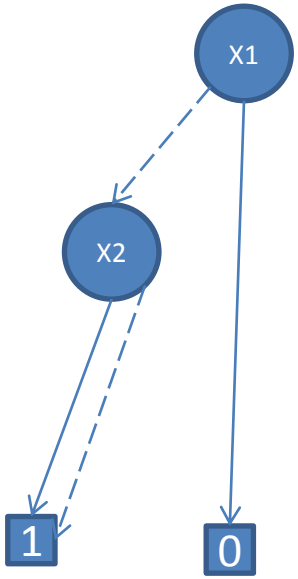
$\exists x'_1 x'_2 . B^1_{AND} =: R^{-1}$

*(OR 0 doesn't change anything)

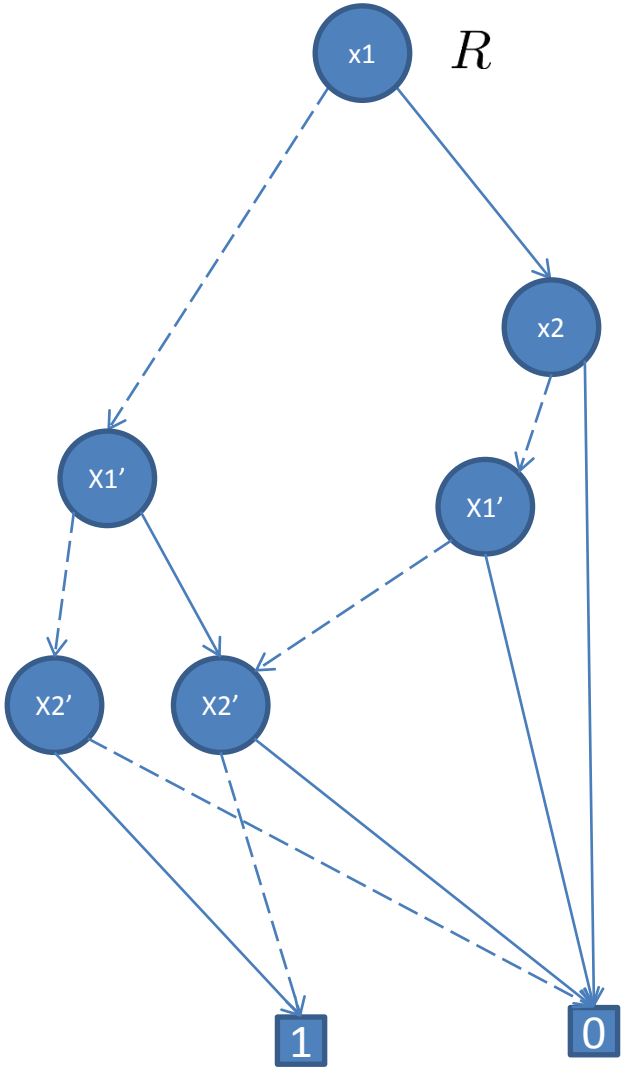
$$[\![\varphi]\!] \quad \cup \quad R^{-1}(Z) \quad := \pi^2(Z)$$





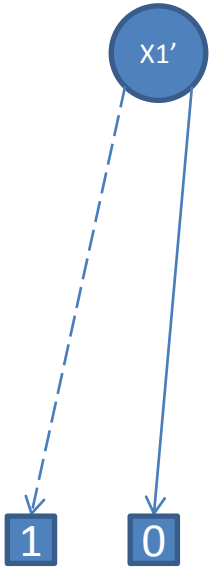


Aufgabe 2

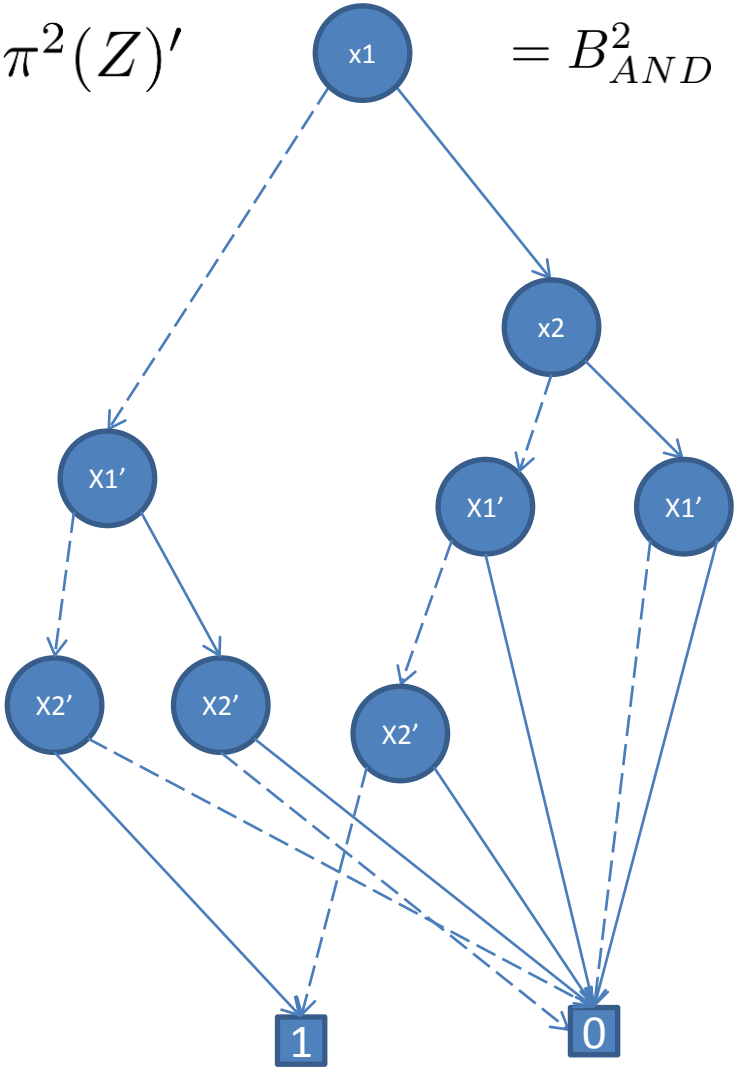


R

AND



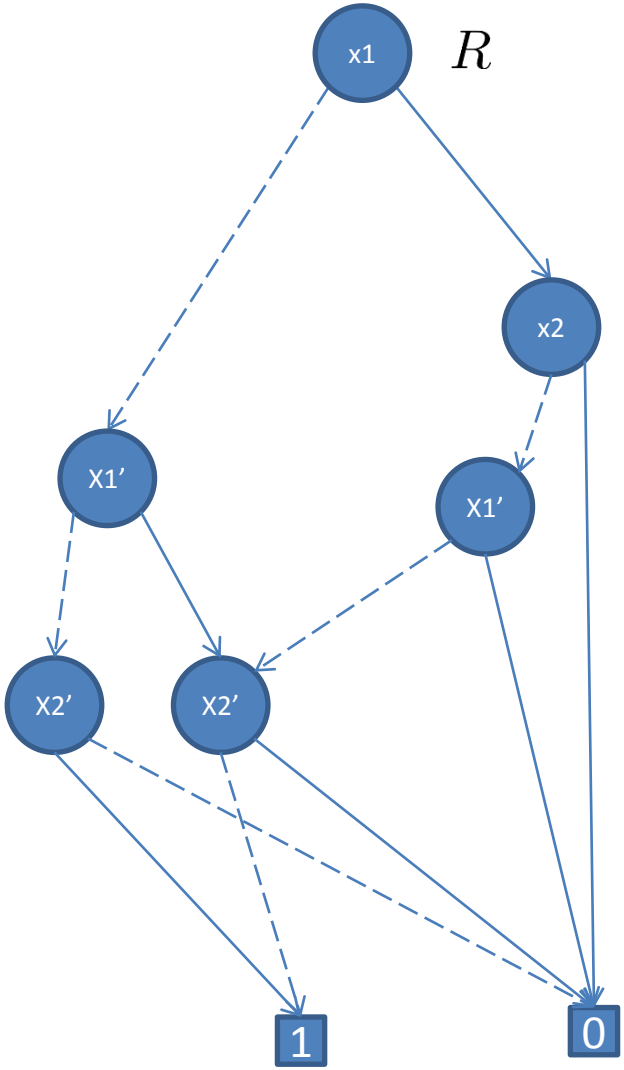
$\pi^2(Z)'$



$= B^2_{AND}$

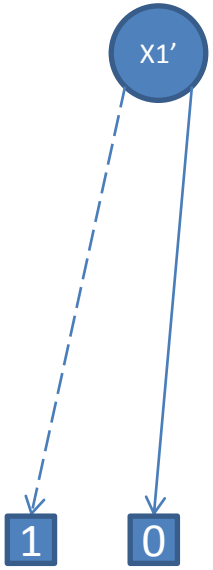
2. Rekursion

Aufgabe 2

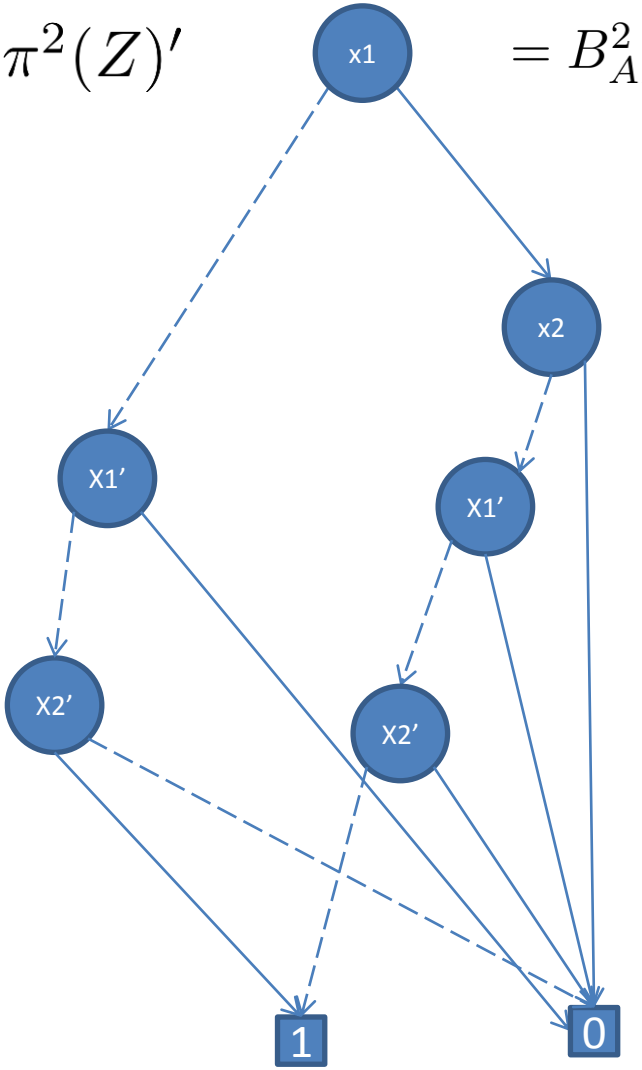


R

AND



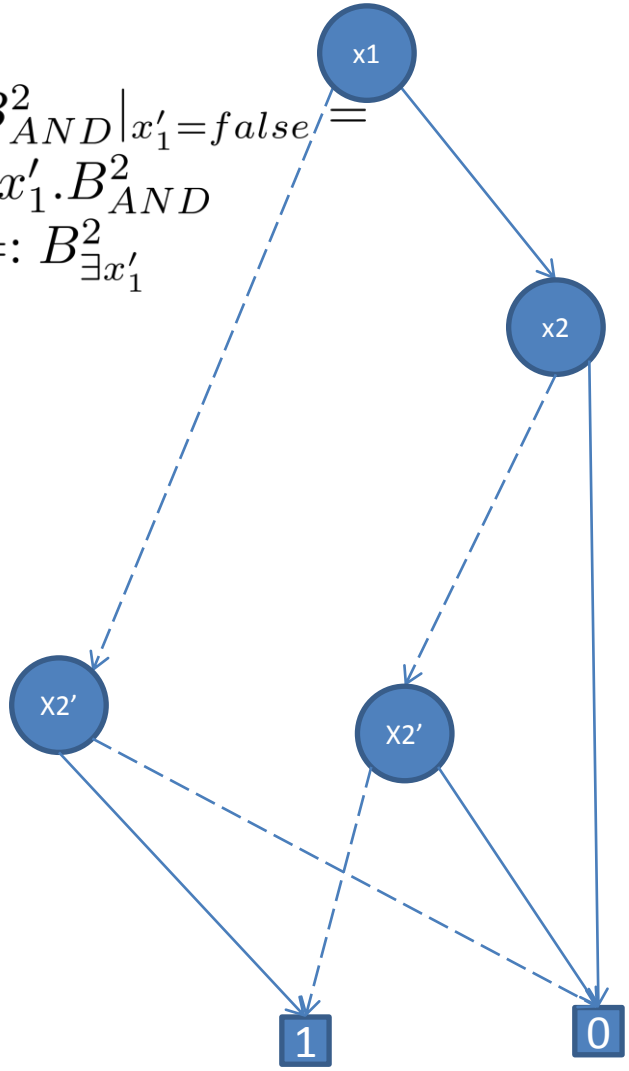
$\pi^2(Z)'$



$= B^2_{AND}$

Aufgabe 2

$B^2_{AND}|_{x'_1=false} = \exists x'_1 . B^2_{AND} =: B^2_{\exists x'_1}$

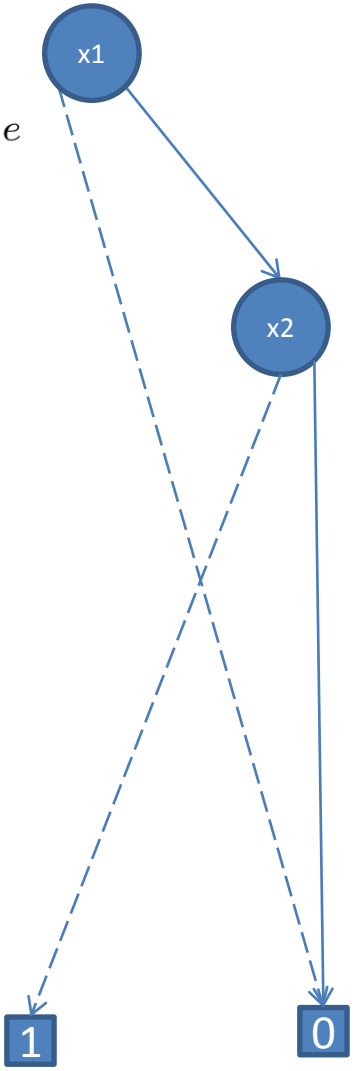


$B^2_{AND}|_{x'_1=true}$

0

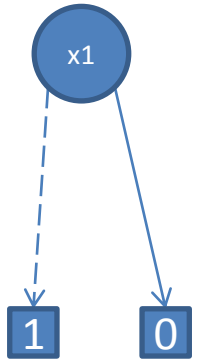
Aufgabe 2

$B^2_{\exists x'_1}|_{x'_2=false}$

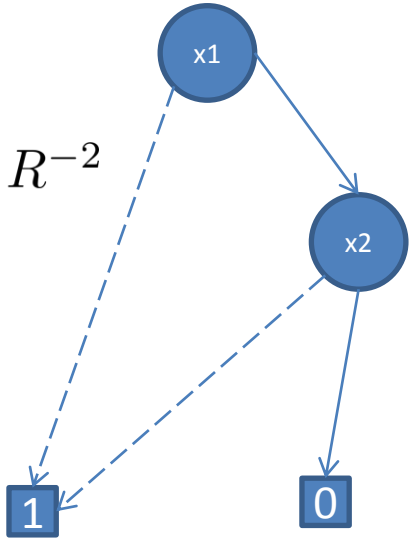


OR

$B^2_{\exists x'_1}|_{x'_2=true}$



$\exists x'_1 x'_2 . B^2_{AND} =: R^{-2}$



Aufgabe 2

AND SO ON…

Man überprüft z.B. leicht, dass R^{-1} gerade die Vorgänger der States beschreibt, die im ersten Schritt die Formed φ erfüllen:

$$\varphi := \neg x_1 \wedge x_2$$

$$[\![\varphi]\!] = \{s_1\}$$

$$R^{-1}([\![\varphi]\!]) = [\![\neg x_1]\!] = \{s_2, s_1\}$$

Logic in Computer Science: Modelling and Reasoning about Systems

Michael Huth, Mark Ryan

Cambridge University Press; Auflage: 2 (26. August 2004)

ISBN-10: 052154310X **ISBN-13:** 978-0521543101