
Principles of the Spin Model Checker

Mordechai (Moti) Ben-Ari

Copyright Springer-Verlag London Limited, 2008

Figure 1.1: Numeric data types in Promela

Type	Values	Size (bits)
bit, bool	0, 1, false, true	1
byte	0..255	8
short	-32768..32767	16
int	$-2^{31}..2^{31} - 1$	32
unsigned	$0..2^n - 1$	≤ 32

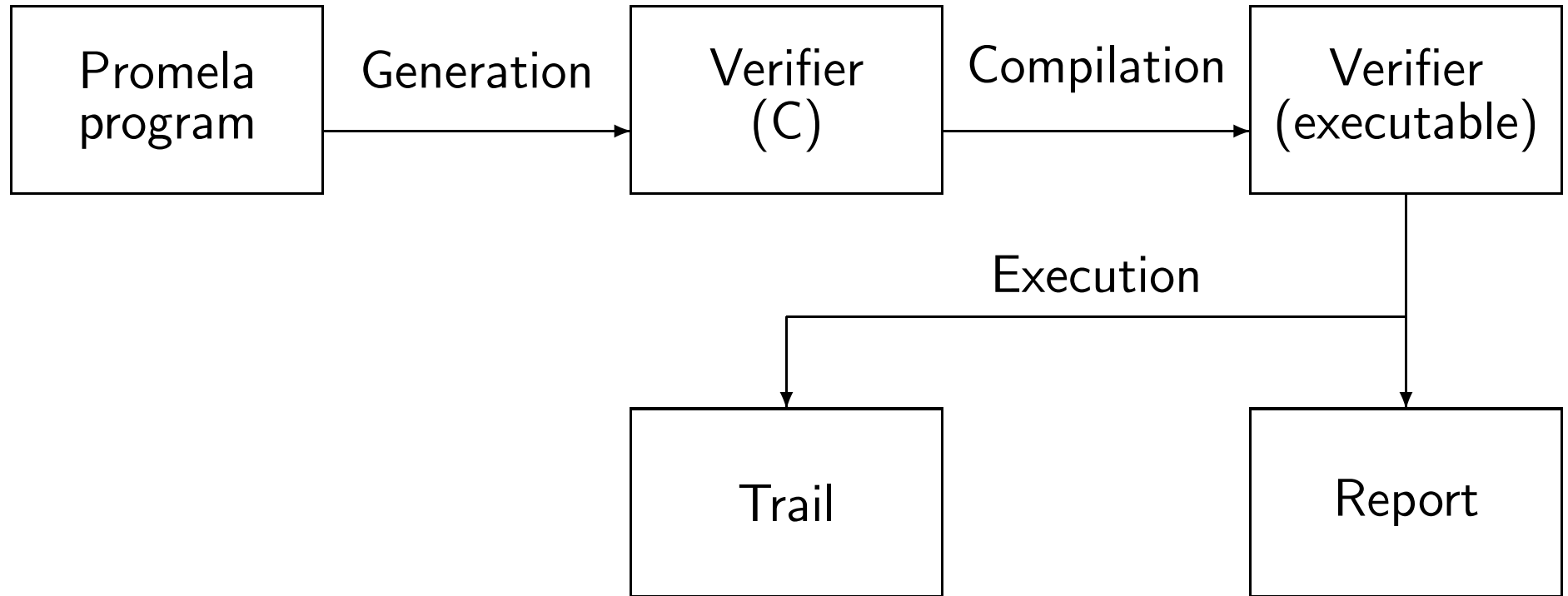
Figure 1.2: Operators in Promela

Precedence	Operator	Associativity	Name
14	()	left	parentheses
14	[]	left	array indexing
14	.	left	field selection
13	!	right	logical negation
13	~	right	bitwise complementation
13	++, --	right	increment, decrement
12	*, /, %	left	multiplication, division, modulo
11	+, -	left	addition, subtraction
10	<<, >>	left	left and right bitwise shift

Figure 1.2: Operators in Promela

Precedence	Operator	Associativity	Name
9	<, <=, >, >=	left	arithmetic relational operators
8	==, !=	left	equality, inequality
7	&	left	bitwise and
6	^	left	bitwise exclusive or
5		left	bitwise inclusive or
4	&&	left	logical and
3		left	logical or
2	(-> :)	right	conditional expression
1	=	right	assignment

Figure 2.1: The architecture of Spin



Computations

Process	Statement	n	Output
P	n = 1	0	
P	printf(P)	1	
Q	n = 2	1	P, n = 1
Q	printf(Q)	2	Q, n = 2

1	2	3	4	5	6
n = 1 printf(P) n = 2 printf(Q)	n = 1 n = 2 printf(P) printf(Q)	n = 1 n = 2 printf(Q) printf(P)	n = 2 printf(Q) n = 1 printf(P)	n = 2 n = 1 printf(Q) printf(P)	n = 2 n = 1 printf(P) printf(Q)

Figure 3.1: Perfect interleaving

Process	Statement	n	P:temp	Q:temp	Output
P	temp = n + 1	0	0	0	
Q	temp = n + 1	0	1	0	
P	n = temp	0	1	1	
Q	n = temp	1	1	1	
P	printf(P)	1	1	1	
Q	printf(Q)	1	1	1	P, n = 1 Q, n = 1

Computation with d_step

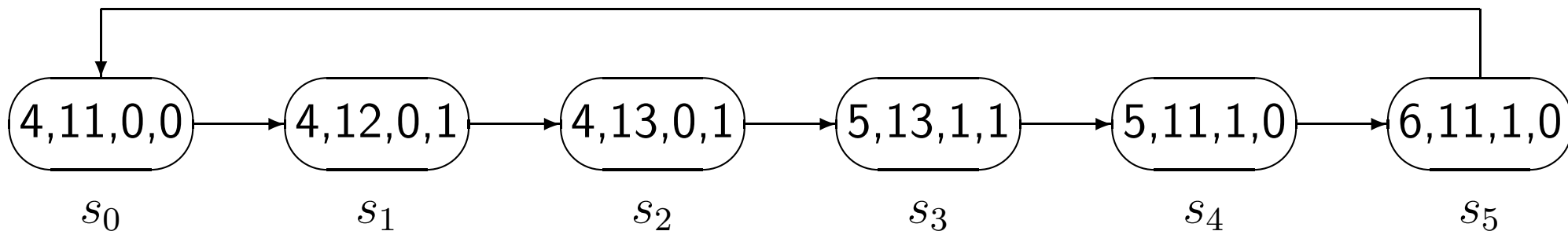
Process	Statement	n	P:temp	Q:temp	Output
P	temp = n+1; n = temp	0	0	0	
Q	temp = n+1; n = temp	1	1	0	
P	printf("P")	2	1	2	
Q	printf("Q")	2	1	2	P, n = 2 Q, n = 2

LTL operators

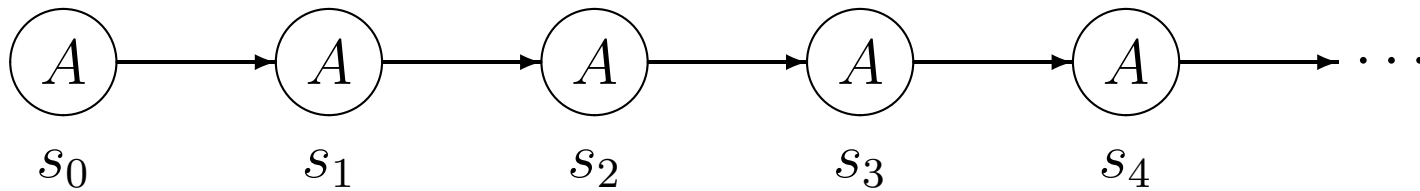
Operator	Math	SPIN
not	\neg	!
and	\wedge	&&
or	\vee	
implies	\rightarrow	->
equivalent	\leftrightarrow	<->

An infinite and a finitely presented computation

$s_0 = (4. \text{ wantP}=1, 11. \text{ wantQ}=1, 0, 0) \longrightarrow$
 $s_1 = (4. \text{ wantP}=1, 12. \text{ !wantP}, 0, 1) \longrightarrow$
 $s_2 = (4. \text{ wantP}=1, 13. \text{ wantQ}=0, 0, 1) \longrightarrow$
 $s_3 = (5. \text{ !wantQ}, 13. \text{ wantQ}=0, 1, 1) \longrightarrow$
 $s_4 = (5. \text{ !wantQ}, 11. \text{ wantQ}=1, 1, 0) \longrightarrow$
 $s_5 = (6. \text{ wantP}=0, 11. \text{ wantQ}=1, 1, 0) \longrightarrow$
 $s_6 = (4. \text{ wantP}=1, 11. \text{ wantQ}=1, 0, 0)$

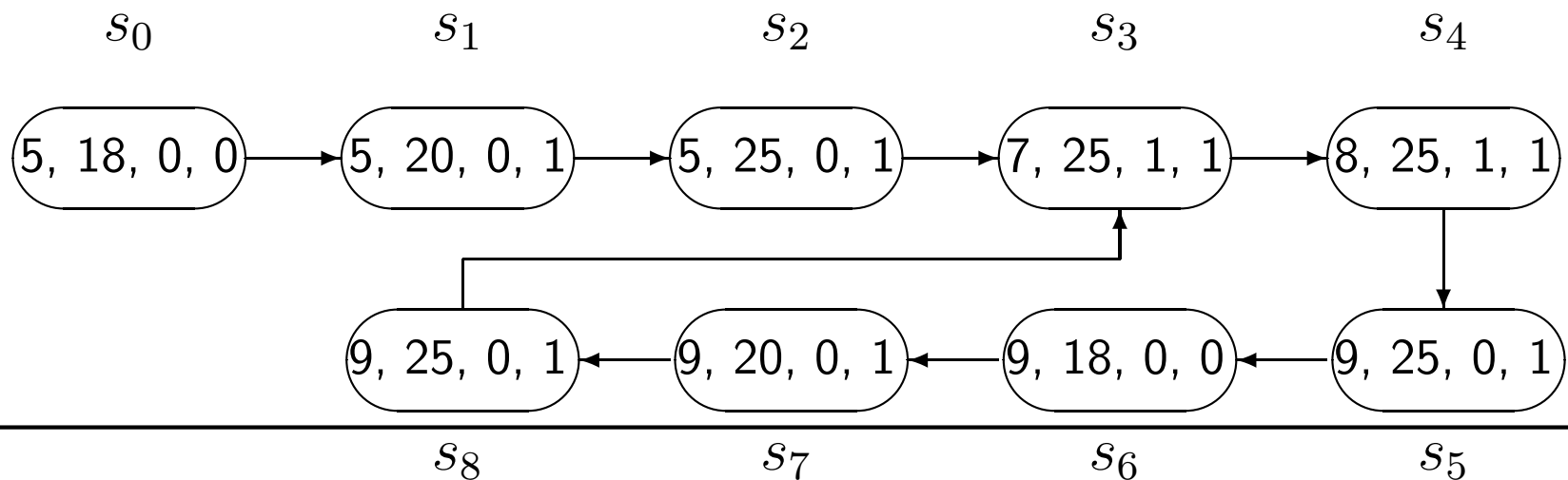


Computation in which $\Box A$ is true



A computation with starvation

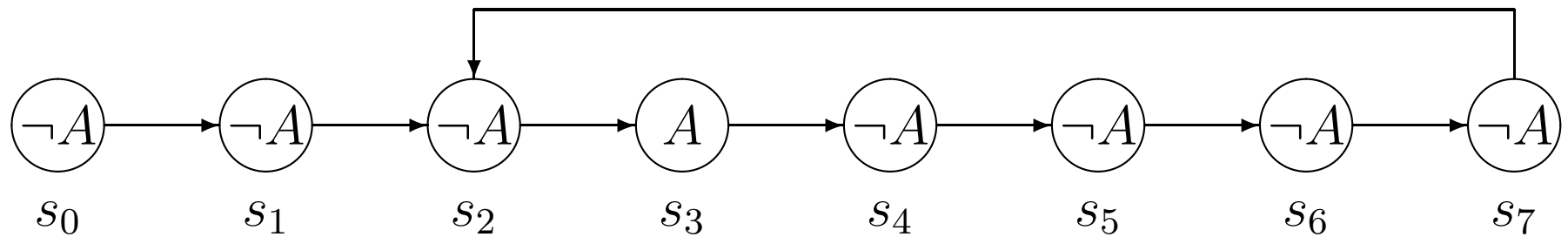
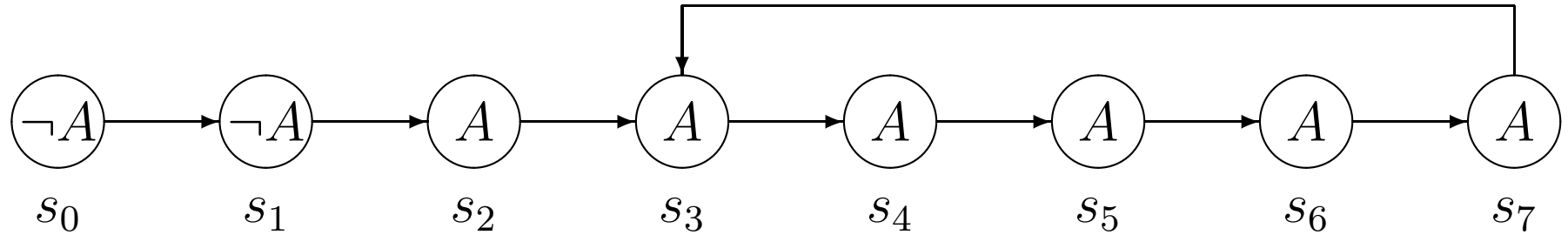
$s_0 = (5. \text{ wantP}=1, 18. \text{ wantQ}=1, 0, 0) \longrightarrow$
 $s_1 = (5. \text{ wantP}=1, 20. \text{ wantP}, 0, 1) \longrightarrow$
 $s_2 = (5. \text{ wantP}=1, 25. \text{ wantQ}=0, 0, 1) \longrightarrow$
 $s_3 = (7. \text{ wantQ}, 25. \text{ wantQ}=0, 1, 1) \longrightarrow$
 $s_4 = (8. \text{ wantP}=0, 25. \text{ wantQ}=0, 1, 1) \longrightarrow$
 $s_5 = (9. \text{ wantP}=1, 25. \text{ wantQ}=0, 0, 1) \longrightarrow$
 $s_6 = (9. \text{ wantP}=1, 18. \text{ wantQ}=1, 0, 0) \longrightarrow$
 $s_7 = (9. \text{ wantP}=1, 20. \text{ wantP}, 0, 1) \longrightarrow$
 $s_8 = (9. \text{ wantP}=1, 25. \text{ wantQ}=0, 0, 1) \longrightarrow$
 $s_9 = (7. \text{ wantQ}, 25. \text{ wantQ}=0, 1, 1)$



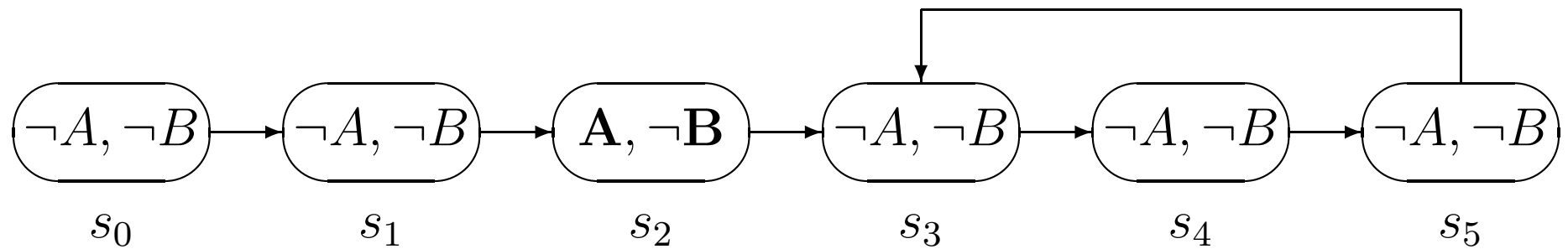
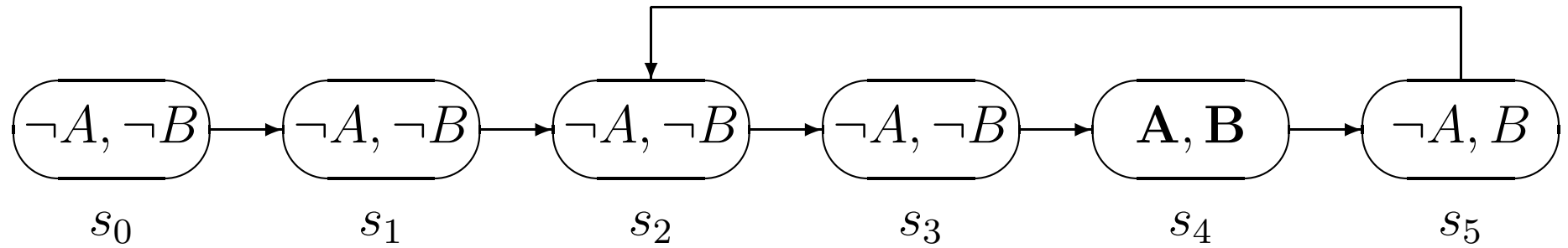
Fairness

$s_0 = (5. \text{ wantP}=1, 18. \text{ wantQ}=1, 0, 0) \longrightarrow$
 $s_1 = (5. \text{ wantP}=1, 20. \text{ wantP}, 0, 1) \longrightarrow$
 $s_2 = (5. \text{ wantP}=1, 25. \text{ wantQ}=0, 0, 1) \longrightarrow$
 $s_3 = (5. \text{ wantP}=1, 18. \text{ wantQ}=1, 0, 0)$

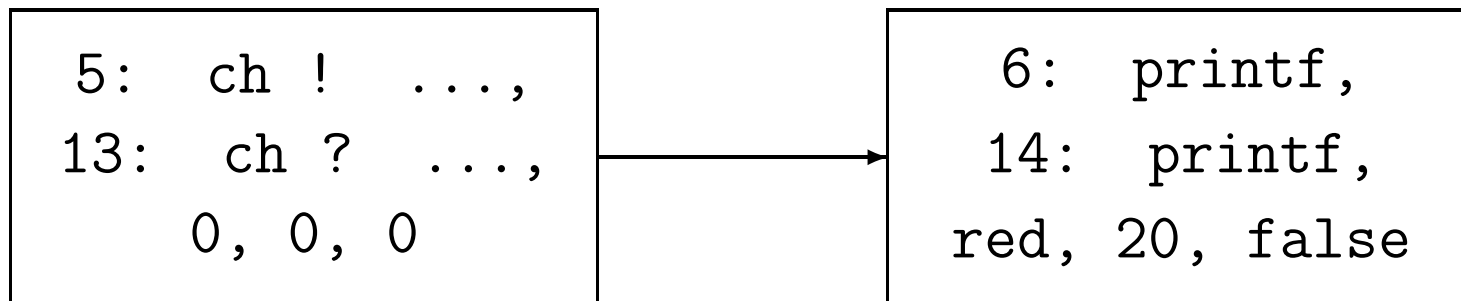
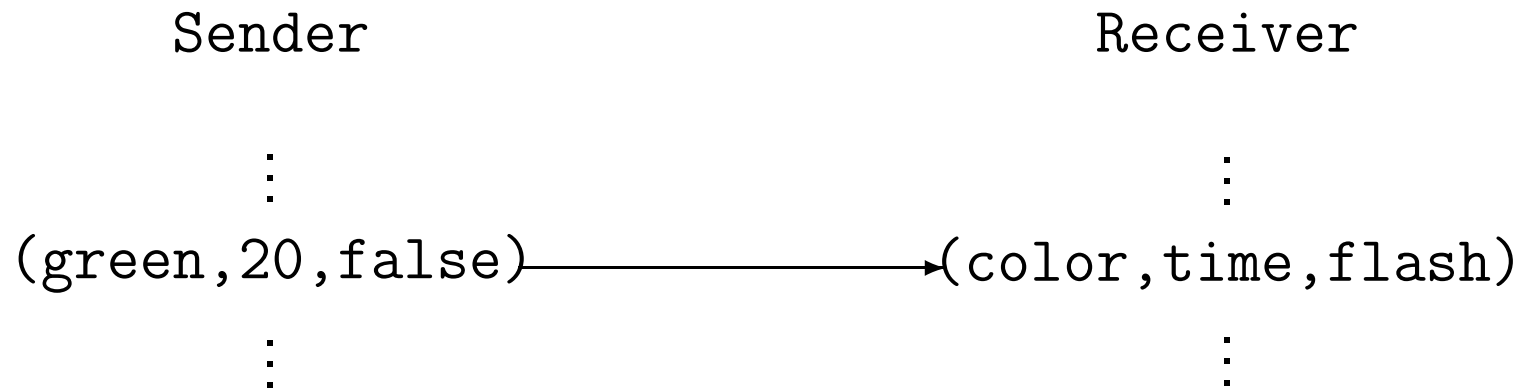
Latching $\Diamond\Box A$ and infinitely often $\Box\Diamond A$



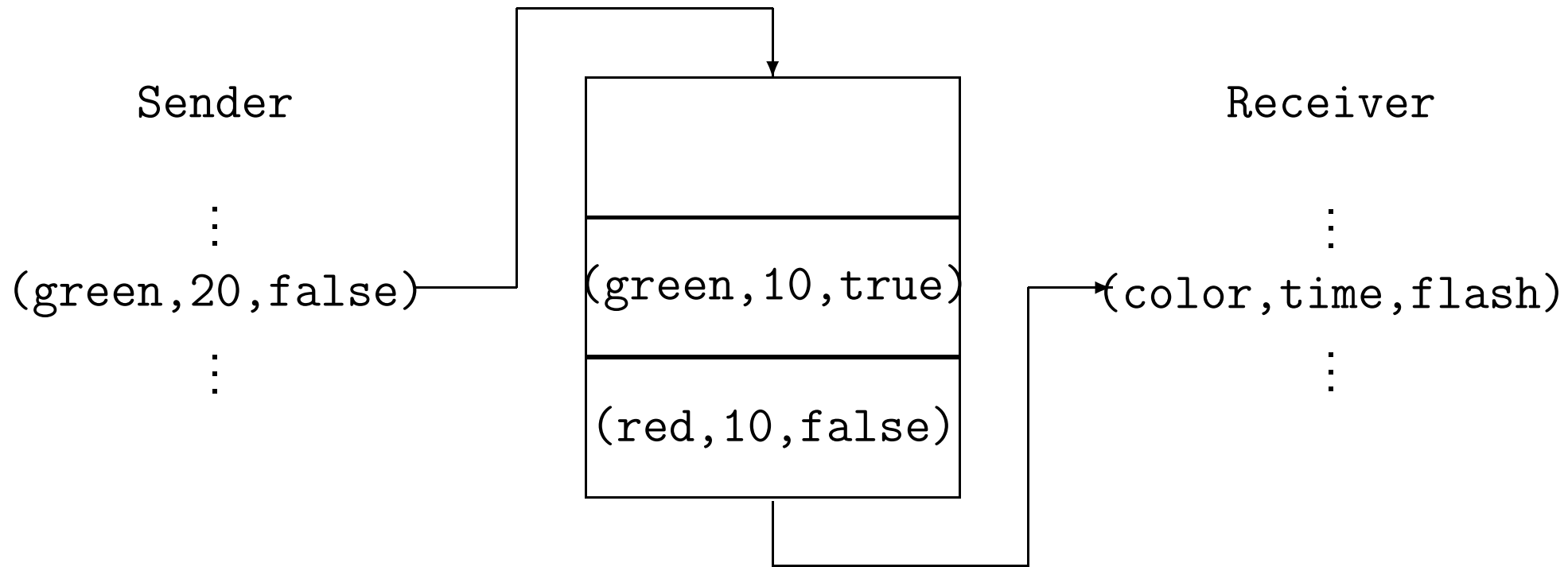
Precedence $\neg B \mathcal{U} A$



Rendezvous channels



Buffered channels



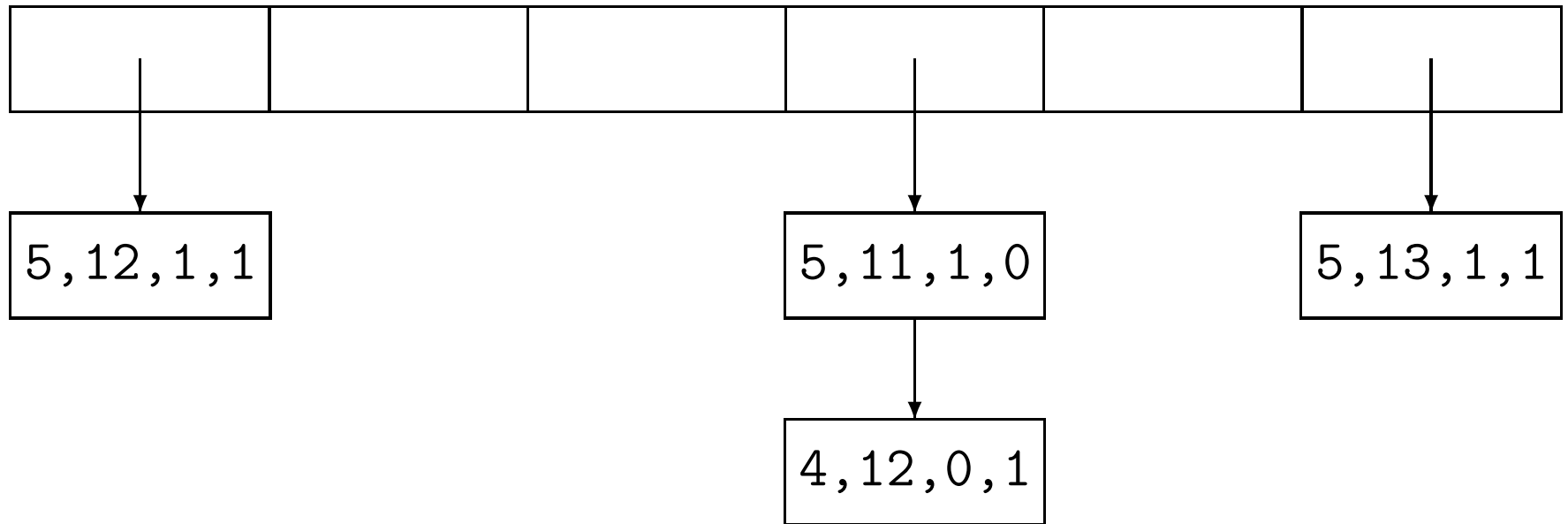
Buffered channels

```
5:  ch !  ...,  
13: ch ?  ...,  
    0, 0, 0,  
[(red,10,false),  
(green,10,true),  
  ()]
```

```
6:  printf,  
13: ch ?  ...,  
    0, 0, 0,  
[(red,10,false),  
(green,10,true),  
(green,20,false)]
```

```
6:  printf,  
14: printf,  
    red, 10, false,  
[(green,10,true),  
(green,20,false),  
  ()]
```

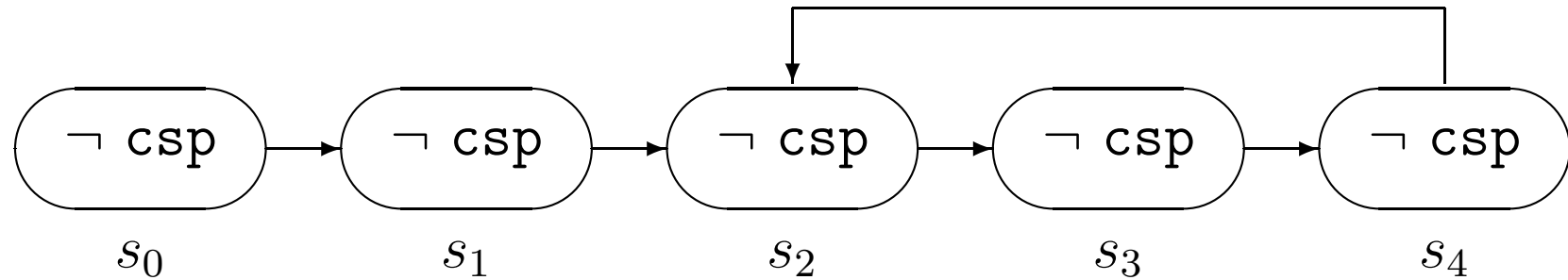
Hash table of state vectors



Effect of hash table size on performance

Hash table (2^w)	Memory (MB)	Conflicts ($\times 10^6$)	Time (sec)
18	270	658	86
20	273	151	33
22	286	52	22
24	336	29	18
26	537	0.1	17

A finitely presented infinite falsifying computation



Adding sparse arrays with channels

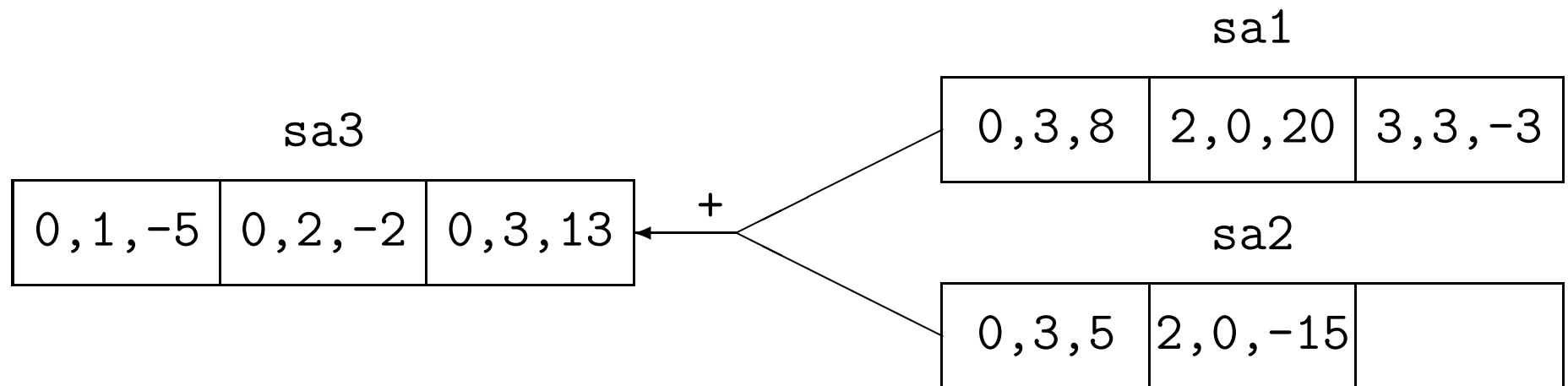
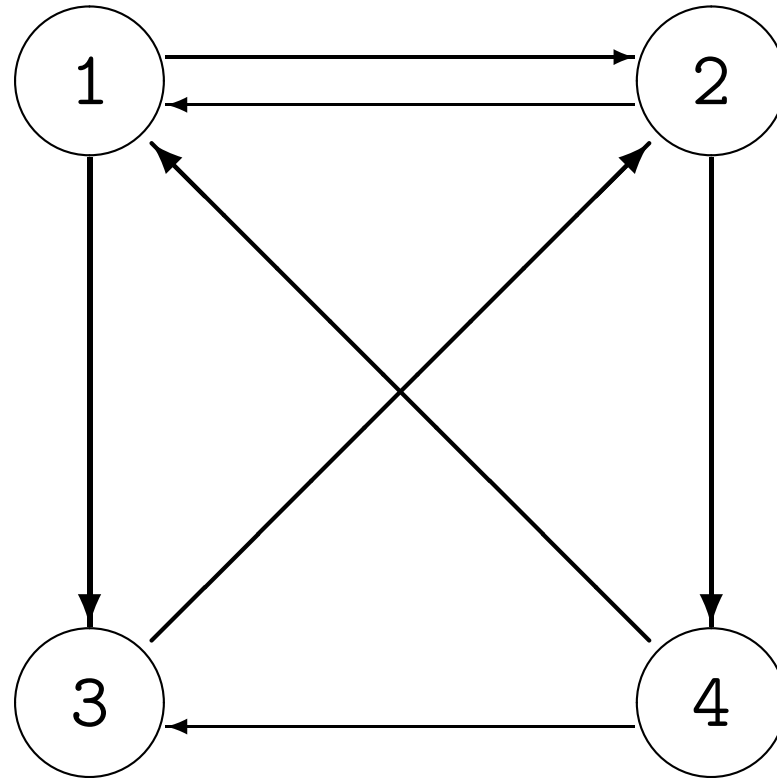


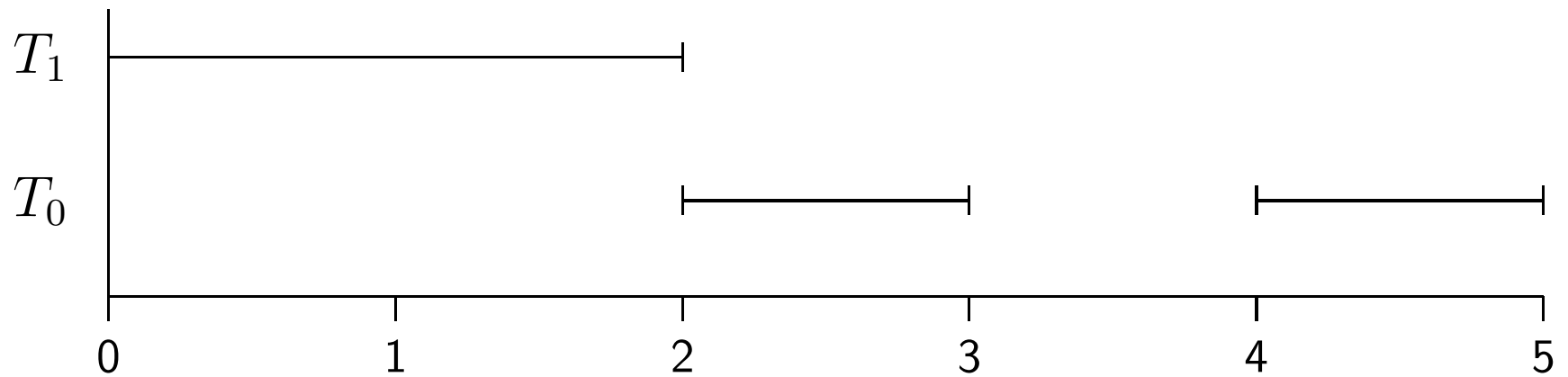
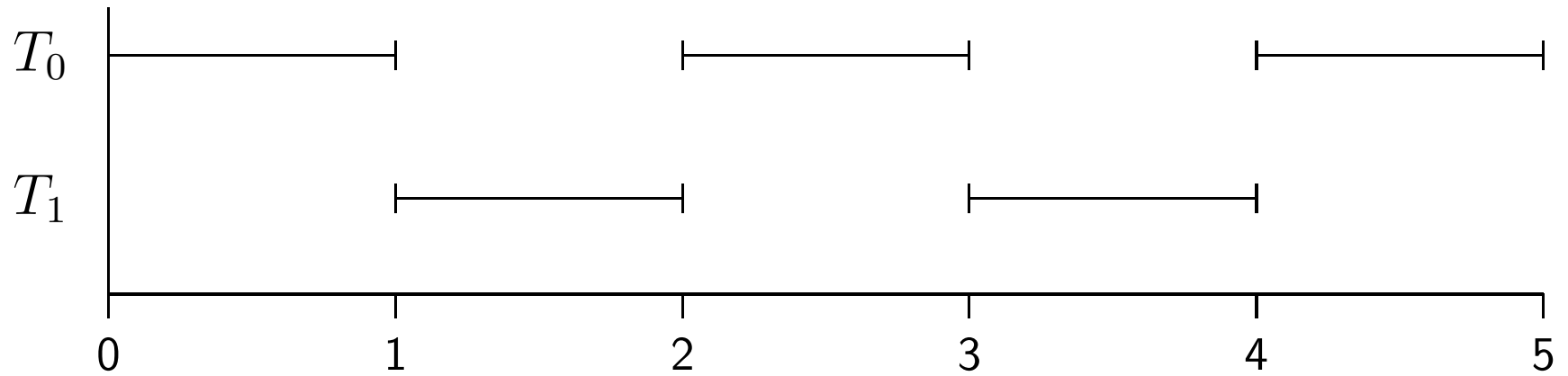
Figure 11.1: A solution to the eight-queens problem

1	Q							
2							Q	
3					Q			
4								Q
5		Q						
6				Q				
7						Q		
8			Q					
	1	2	3	4	5	6	7	8

Figure 11.2: Simple cycles in a directed graph



Real-time systems



The development of Fischer's algorithm

Process	Statement	turn
P1	turn == 0	0
P2	turn == 0	0
P1	turn = 1	0
P1	turn == 1	1
P1	/* CS */	1
P2	turn = 2	1
P2	turn == 2	2
P2	/* CS */	2

Process	Statement	turn
P1	turn == 0	0
P1	turn = 1	0
P2	turn == 0	1
P1	turn == 1	1

Messages in a channel

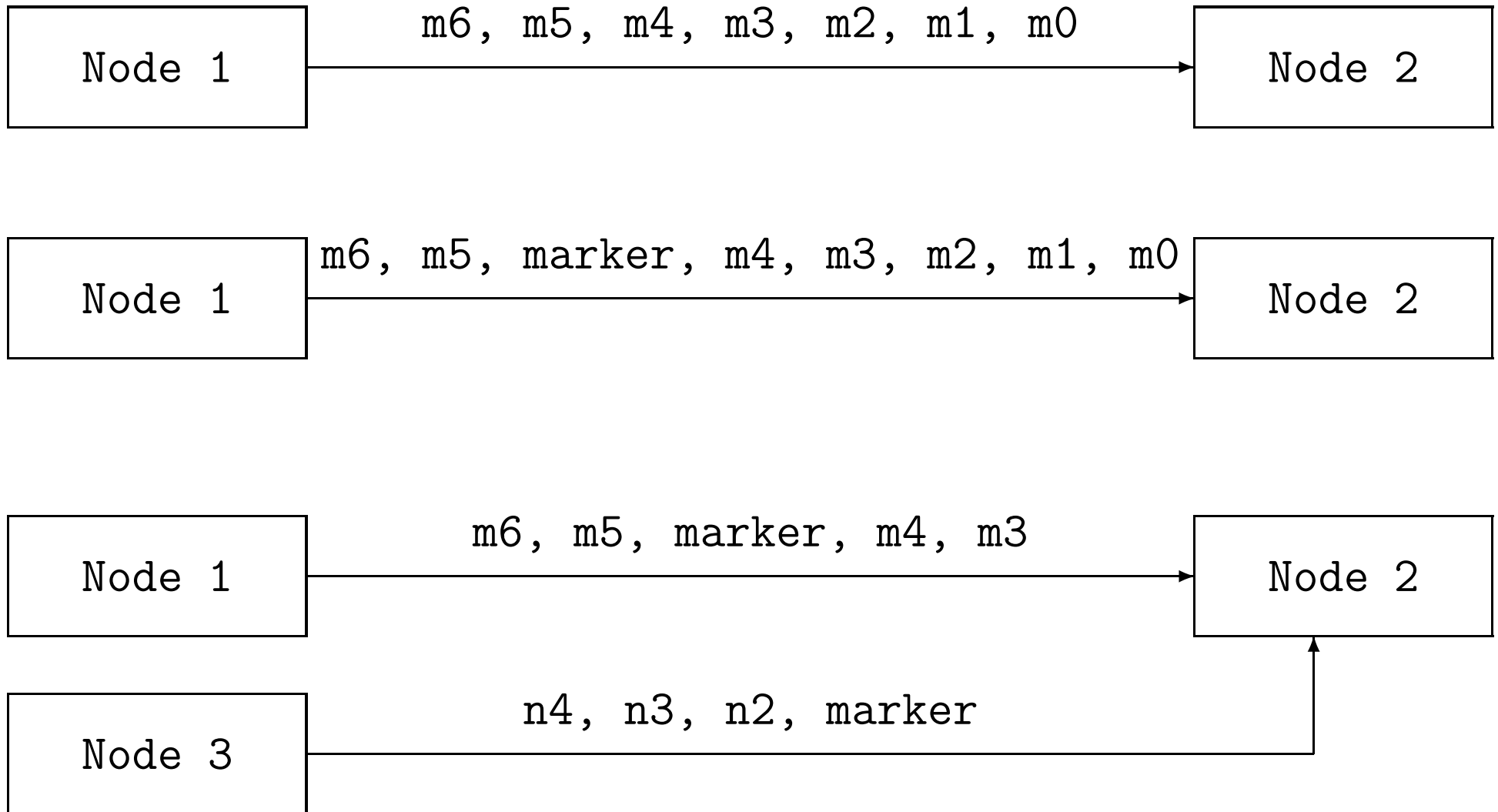
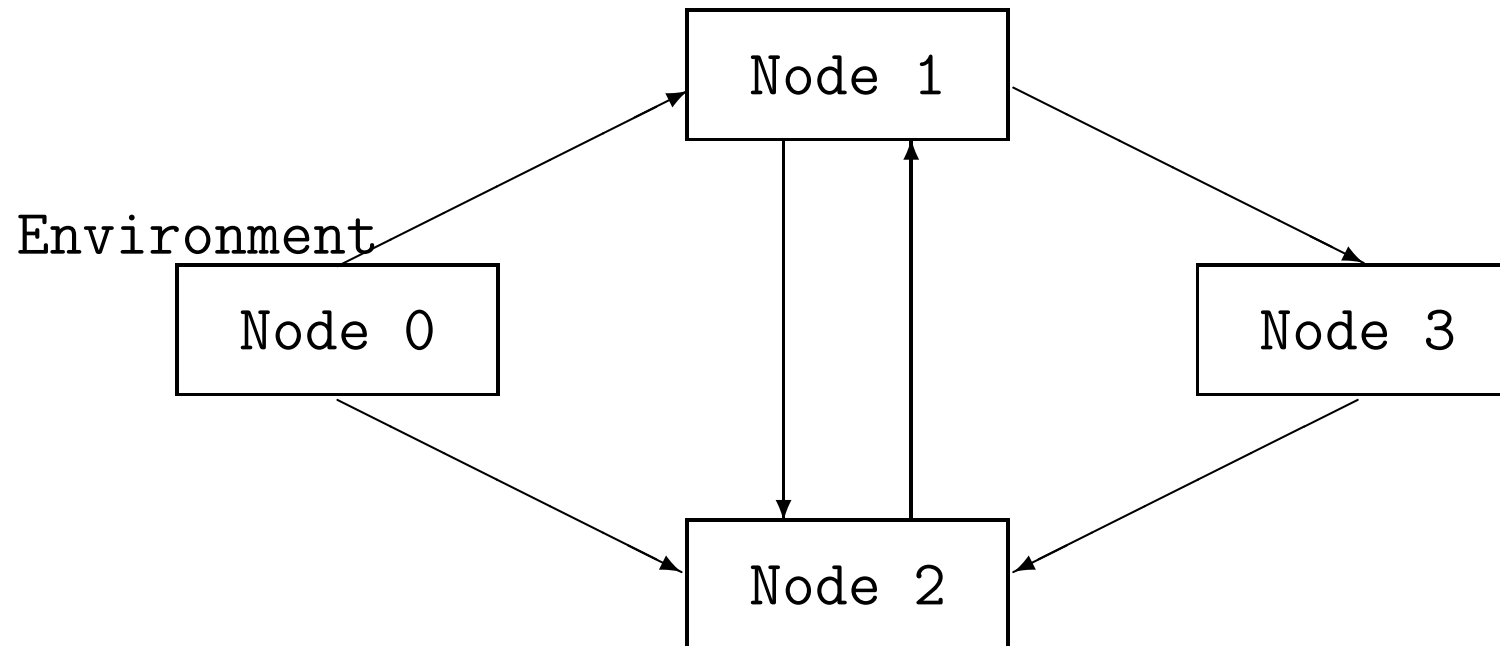


Figure 11.3: A directed graph of nodes



Encoding lists of channels and channel capacity

Incoming

0	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

Outgoing

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

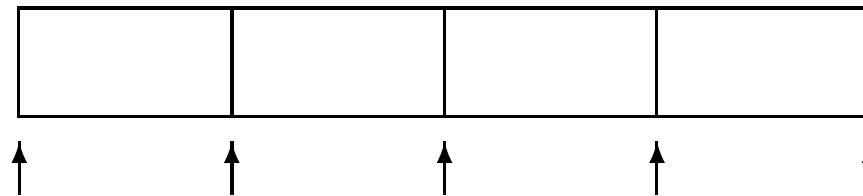


Figure A.2: The structure of SpinSpider

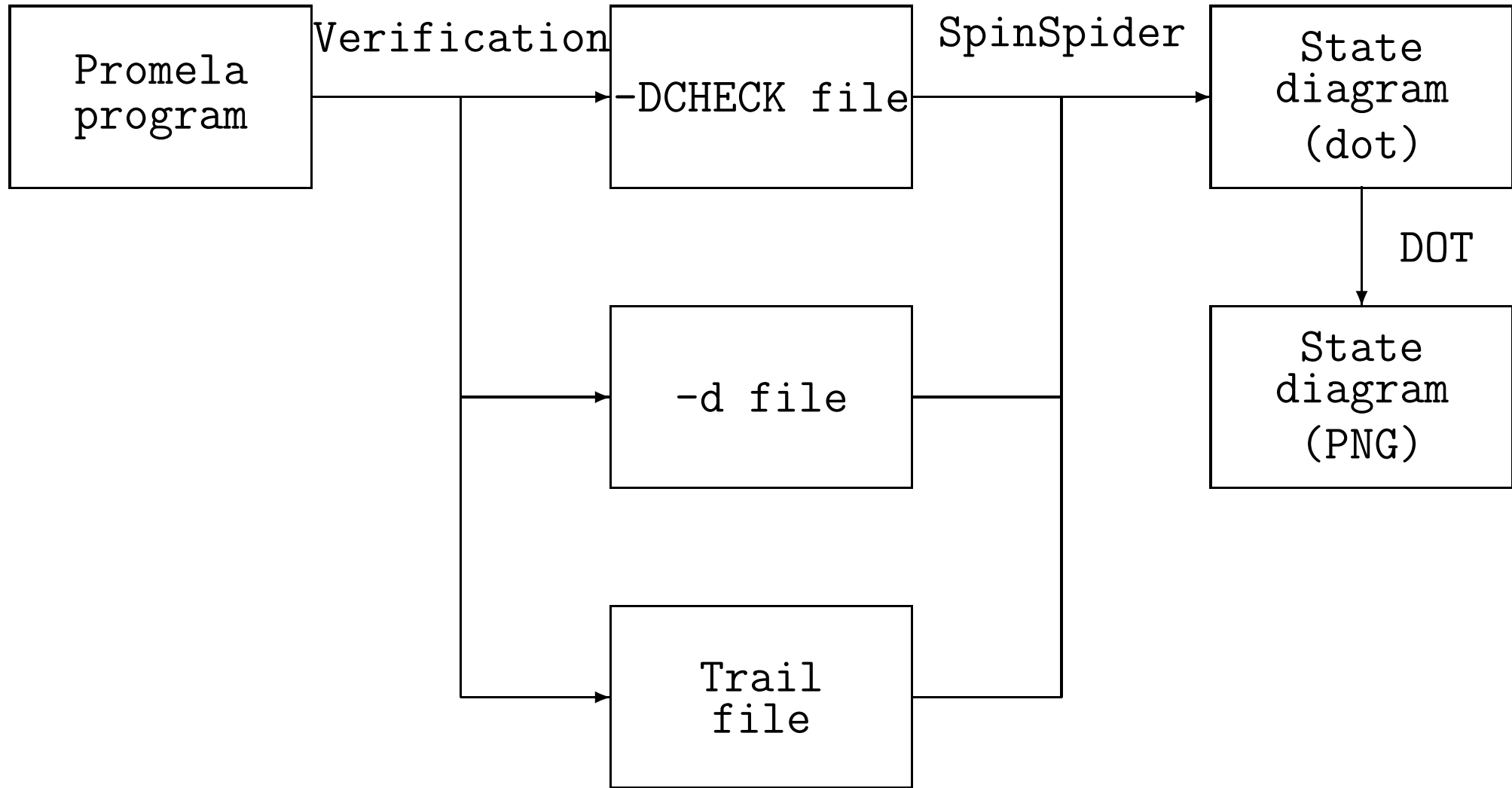


Figure A.4: The structure of VN

