Ein Entscheidungsverfahren für Presburger-Arithmetik

Fabio Madge Pimentel

Technische Universität München

05. Juli 2016

1 Grundlagen

1 Grundlagen

2 Automatenkonstruktion

1 Grundlagen

2 Automatenkonstruktion

3 Ganze Zahlen

1 Grundlagen
- Syntax
- Beispiel
- Theorie

2 Automatenkonstruktion

3 Ganze Zahlen

1 Grundlagen
- Syntax
- Beispiel
- Theorie

2 Automatenkonstruktion

3 Ganze Zahlen

$$\langle \textit{formula} \rangle ::= \text{`}\neg\text{'}\ \langle \textit{formula} \rangle$$
$$\mid \quad \langle \textit{formula} \rangle\ \text{`}\vee\text{'}\ \langle \textit{formula} \rangle$$
$$\mid \quad \text{`}\exists\text{'}\ \langle \textit{var} \rangle\ \langle \textit{formula} \rangle$$
$$\mid \quad \langle \textit{atomicformula} \rangle$$

$$\langle \textit{atomicformula} \rangle ::= \langle \textit{term} \rangle\ \text{`}=\text{'}\ \langle \textit{term} \rangle$$

$$\langle \textit{term} \rangle ::= \langle \textit{term} \rangle\ \text{`}+\text{'}\ \langle \textit{term} \rangle$$
$$\mid \quad \langle \textit{variable} \rangle$$
$$\mid \quad \langle \textit{constant} \rangle$$

$$\langle \textit{variable} \rangle ::= \mathsf{x} \mid \mathsf{y} \mid \mathsf{z} \mid \ldots$$

$$\langle \textit{constant} \rangle ::= 0 \mid 1$$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg \exists x.\neg P(x)$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg\exists x.\neg P(x)$
- $x \neq y ::= \neg(x = y)$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg\exists x.\neg P(x)$
- $x \neq y ::= \neg(x = y)$
- $x \leq y ::= \exists z.(x + z = y)$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg\exists x.\neg P(x)$
- $x \neq y ::= \neg(x = y)$
- $x \leq y ::= \exists z.(x + z = y)$
- $x \geq y ::= y \leq x$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg\exists x.\neg P(x)$
- $x \neq y ::= \neg(x = y)$
- $x \leq y ::= \exists z.(x + z = y)$
- $x \geq y ::= y \leq x$
- $x < y ::= x \leq y \wedge x \neq y$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg\exists x.\neg P(x)$
- $x \neq y ::= \neg(x = y)$
- $x \leq y ::= \exists z.(x + z = y)$
- $x \geq y ::= y \leq x$
- $x < y ::= x \leq y \wedge x \neq y$
- $x > y ::= y < x$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg\exists x.\neg P(x)$
- $x \neq y ::= \neg(x = y)$
- $x \leq y ::= \exists z.(x + z = y)$
- $x \geq y ::= y \leq x$
- $x < y ::= x \leq y \wedge x \neq y$
- $x > y ::= y < x$
- $n \iff \sum_{i=1}^{n} 1 ::= 1 + ... + 1$

- $x \wedge y ::= \neg(\neg x \vee \neg y)$
- $x \rightarrow y ::= \neg x \vee y$
- $\forall x.P(x) ::= \neg\exists x.\neg P(x)$
- $x \neq y ::= \neg(x = y)$
- $x \leq y ::= \exists z.(x + z = y)$
- $x \geq y ::= y \leq x$
- $x < y ::= x \leq y \wedge x \neq y$
- $x > y ::= y < x$
- $n \iff \sum_{i=1}^{n} 1 ::= 1 + ... + 1$
- $nx \iff \sum_{i=1}^{n} x ::= x + ... + x$

1 Grundlagen
- Syntax
- Beispiel
- Theorie

2 Automatenkonstruktion

3 Ganze Zahlen

	interner Bedarf	externer Bedarf
Bolzen	$2G + 3K$	5000
Gelenke	$2K$	2000
Kurbelwellen	0	1000
Lager	$4K$	2500

Tabelle: Badarfsanalyse

	interner Bedarf	externer Bedarf
Bolzen	$2G + 3K$	5000
Gelenke	$2K$	2000
Kurbelwellen	0	1000
Lager	$4K$	2500

Tabelle: Badarfsanalyse

$$
\begin{aligned}
& 5000 + 2G + 3K = B \\
\wedge \quad & 2000 + 2K = G \\
\wedge \quad & 1000 = K \\
\wedge \quad & 2500 + 4K = L
\end{aligned}
$$

1 Grundlagen
- Syntax
- Beispiel
- Theorie

2 Automatenkonstruktion

3 Ganze Zahlen

Axiomatisierung von $Th(\mathbb{N}, +, =)$*(Die freien Variablen sind implizit allquantifiziert)*:

- $\neg(x + 1 = 0)$

Axiomatisierung von $Th(\mathbb{N}, +, =)$*(Die freien Variablen sind implizit allquantifiziert)*:

- $\neg(x + 1 = 0)$
- $(x + 1 = y + 1 \rightarrow x = y)$

Axiomatisierung von $Th(\mathbb{N}, +, =)$*(Die freien Variablen sind implizit allquantifiziert)*:

- $\neg(x + 1 = 0)$
- $(x + 1 = y + 1 \rightarrow x = y)$
- $x + 0 = x$

Axiomatisierung von $Th(\mathbb{N}, +, =)$*(Die freien Variablen sind implizit allquantifiziert)*:

- $\neg(x + 1 = 0)$
- $(x + 1 = y + 1 \rightarrow x = y)$
- $x + 0 = x$
- $(x + y) + 1 = x + (y + 1)$

Axiomatisierung von $Th(\mathbb{N}, +, =)$*(Die freien Variablen sind implizit allquantifiziert)*:

- $\neg(x + 1 = 0)$
- $(x + 1 = y + 1 \rightarrow x = y)$
- $x + 0 = x$
- $(x + y) + 1 = x + (y + 1)$
- $(P(0) \wedge \forall x.P(x) \rightarrow P(x + 1)) \rightarrow \forall x.P(x)$

1 Grundlagen

2 Automatenkonstruktion
- Kodierung
- Formel
- Gleichungen
- Ungleichungen

3 Ganze Zahlen

1 Grundlagen

2 Automatenkonstruktion
- Kodierung
- Formel
- Gleichungen
- Ungleichungen

3 Ganze Zahlen

- $\Sigma = \{0, 1\}^n$, for free variables $x_1, \ldots x_n$.

- $\Sigma = \{0,1\}^n$, for free variables $x_1, \ldots x_n$.
- $L(\varphi) = \bigcup_{s \in Sol(\varphi)} \texttt{LSBF}(s)$

- $\Sigma = \{0,1\}^n$, for free variables $x_1, \ldots x_n$.
- $L(\varphi) = \bigcup_{s \in Sol(\varphi)} \texttt{LSBF}(s)$

Example:

$$\texttt{LSBF}(5,10,0) = \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}\left(\begin{bmatrix}0\\0\\0\end{bmatrix}\right)^* \qquad \begin{aligned} 2^0 + 2^2 &= 5 \\ 2^1 + 2^3 &= 3 \\ 0 &= 0 \end{aligned}$$

1 Grundlagen

2 Automatenkonstruktion
- Kodierung
- Formel
- Gleichungen
- Ungleichungen

3 Ganze Zahlen

```
Function FtoNFA(φ)
    switch φ do
        case ¬φ do
            return CompNFA(FtoNFA(φ));
        case φ1 ∨ φ2 do
            I ← Free(φ) ∪ Free(φ);
            s1 ← Projection(I,FtoNFA(φ1));
            s2 ← Projection(I,FtoNFA(φ2));
            return Union(s1, s2);
        case ∃x.φ do
            I ← Free(φ) − x;
            return Projection(I,FtoNFA(φ));
        otherwise do
            return DFAtoNFA(EqtoDFA(φ));
```

1 Grundlagen

2 Automatenkonstruktion
- Kodierung
- Formel
- Gleichungen
- Ungleichungen

3 Ganze Zahlen

Betrachtung von Gleichungen der Form

$a_1 x_1 + \ldots + a_n x_n = ax = b$

Betrachtung von Gleichungen der Form
$a_1x_1 + \ldots + a_nx_n = ax = b$

Idee:
Jeder Zustand $q \in \mathbb{Z}$ erkennt jene Wörter $c \in \mathbb{N}^n$, für die $ac = q$ gilt.

$q \xrightarrow{\zeta} q'$?

$c' \in L(q')$

$q \xrightarrow{\zeta} q'$?

$$
\begin{aligned}
& c' \in L(q') \\
\iff & 2c' + \zeta \in L(q) && \text{(def LSBF)}
\end{aligned}
$$

$q \xrightarrow{\zeta} q'$?

$$
\begin{aligned}
& c' \in L(q') \\
\iff & 2c' + \zeta \in L(q) && \text{(def LSBF)} \\
\iff & a(2c' + \zeta) = q && \text{(Idee)}
\end{aligned}
$$

$q \xrightarrow{\zeta} q'$?

$$
\begin{aligned}
& c' \in L(q') \\
\iff & 2c' + \zeta \in L(q) && \text{(def LSBF)} \\
\iff & a(2c' + \zeta) = q && \text{(Idee)} \\
\iff & ac' = \frac{1}{2}(q - a\zeta)
\end{aligned}
$$

$q \xrightarrow{\zeta} q'$?

$$
\begin{aligned}
& c' \in L(q') \\
\iff & 2c' + \zeta \in L(q) && \text{(def LSBF)} \\
\iff & a(2c' + \zeta) = q && \text{(Idee)} \\
\iff & ac' = \frac{1}{2}(q - a\zeta)
\end{aligned}
$$

$$
\delta(q, \zeta) = \begin{cases} q_t & \text{if } q = q_t \text{ or } q - a\zeta \text{ is odd} \\ \frac{1}{2}(q - a\zeta) & \text{if } q - a\zeta \text{ is even} \end{cases}
$$

`EqtoDFA(`φ`)`

Data : Equation $\varphi ::= ax = b$

Result : DFA $A_\varphi = (Q, \Sigma, \delta, q_0, F)$ such that $L(A_\varphi) = L(\varphi)$

$Q, \delta, F \leftarrow \emptyset$;

$q_0 \leftarrow s_b$;

$W \leftarrow \{s_b\}$;

while $W \neq \emptyset$ **do**

  $s_k \leftarrow$ `pick(`W`)`;

  $Q \leftarrow Q \cup \{s_k\}$;

  if $k = 0$ **then** $F \leftarrow F \cup \{s_k\}$;

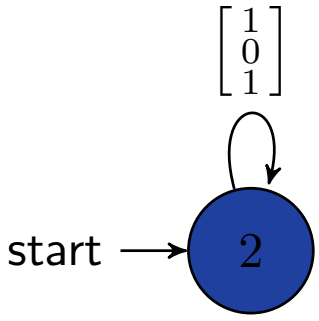
  forall $\zeta \in \{0, 1\}^n$ **do**

    if $k - a\zeta$ *is even* **then**

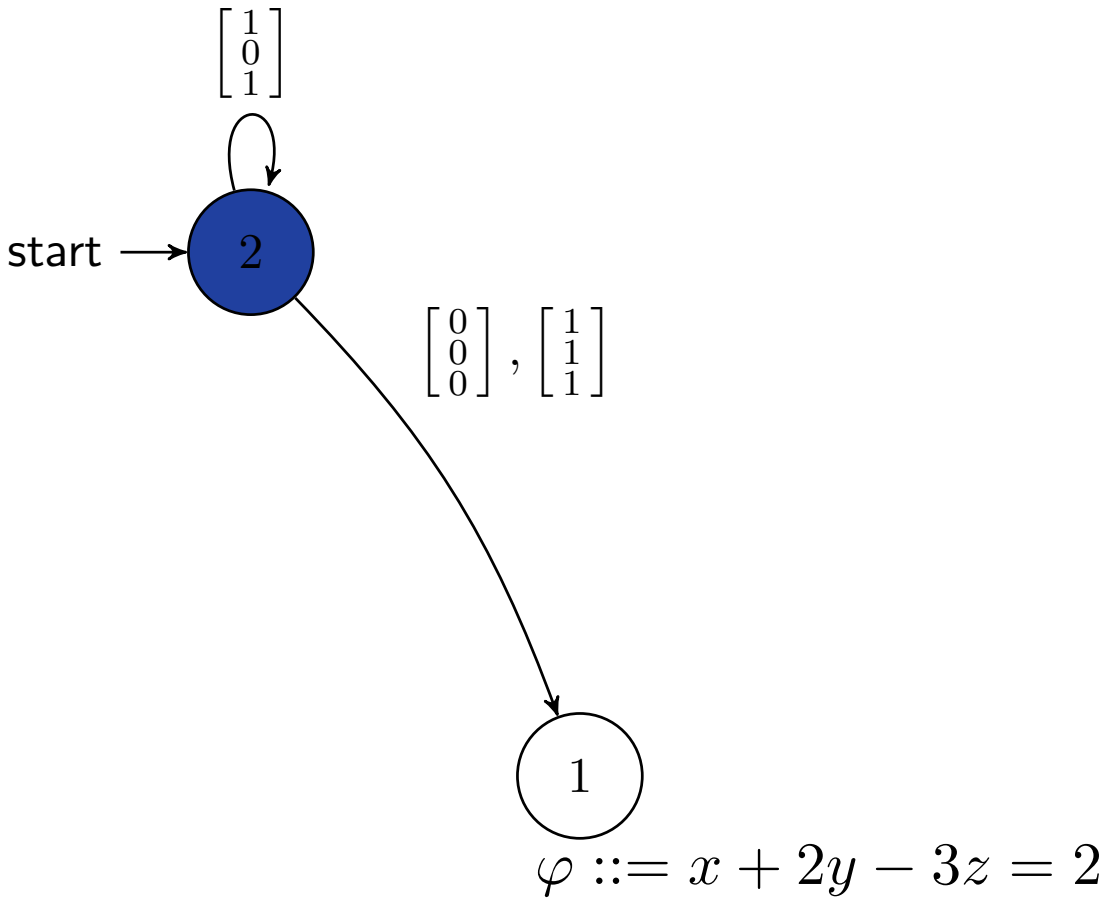
      $j \leftarrow \frac{1}{2}(k - a\zeta)$;

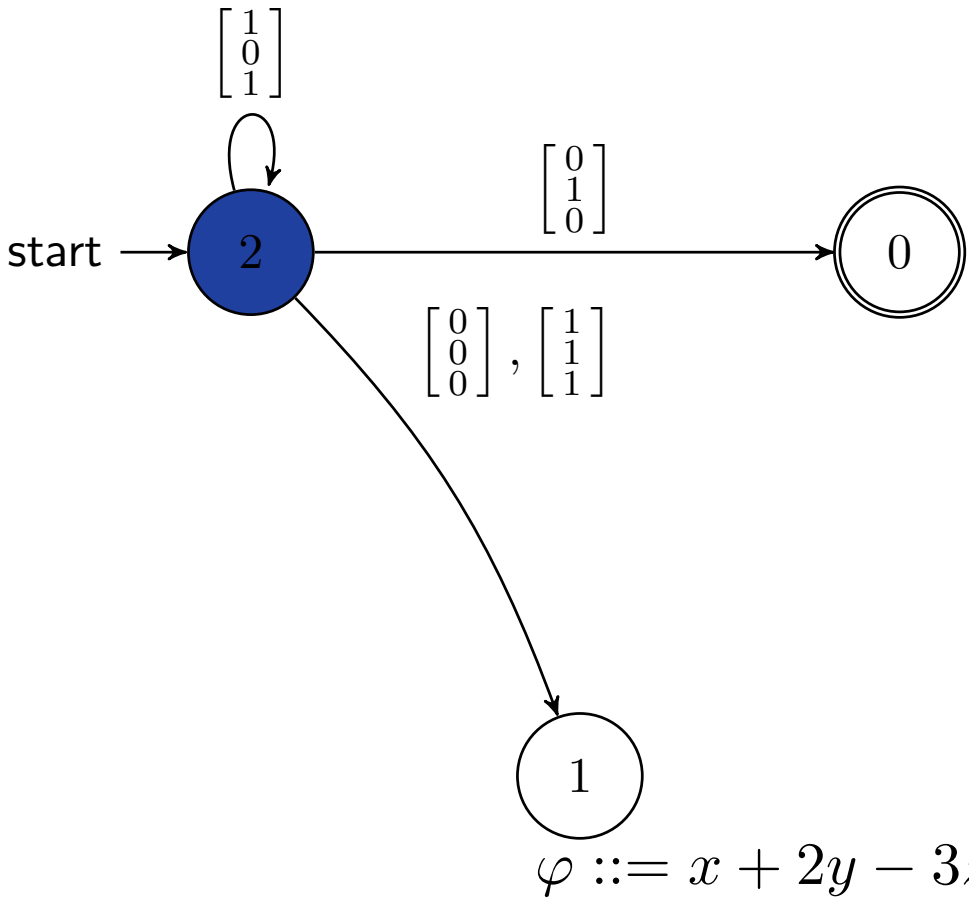
      if $s_j \notin Q$ **then** $W \leftarrow W \cup \{s_j\}$;

      $\delta \leftarrow \delta \cup (s_k, \zeta, s_j)$;

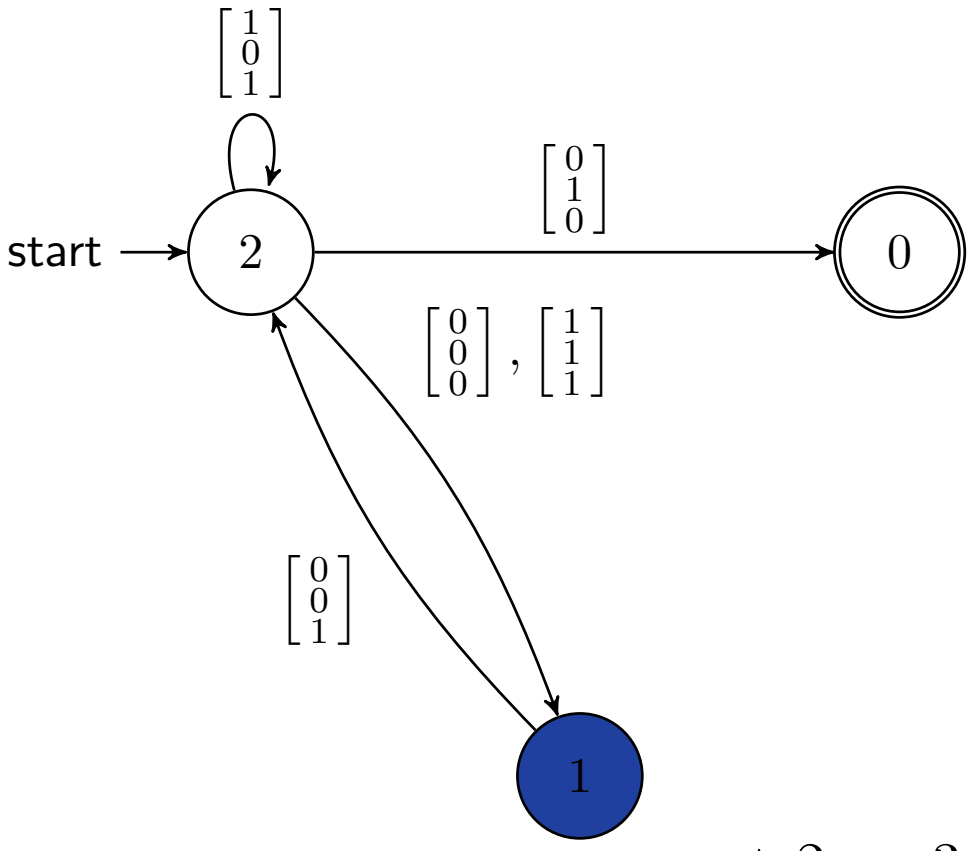


$$\varphi ::= x + 2y - 3z = 2$$

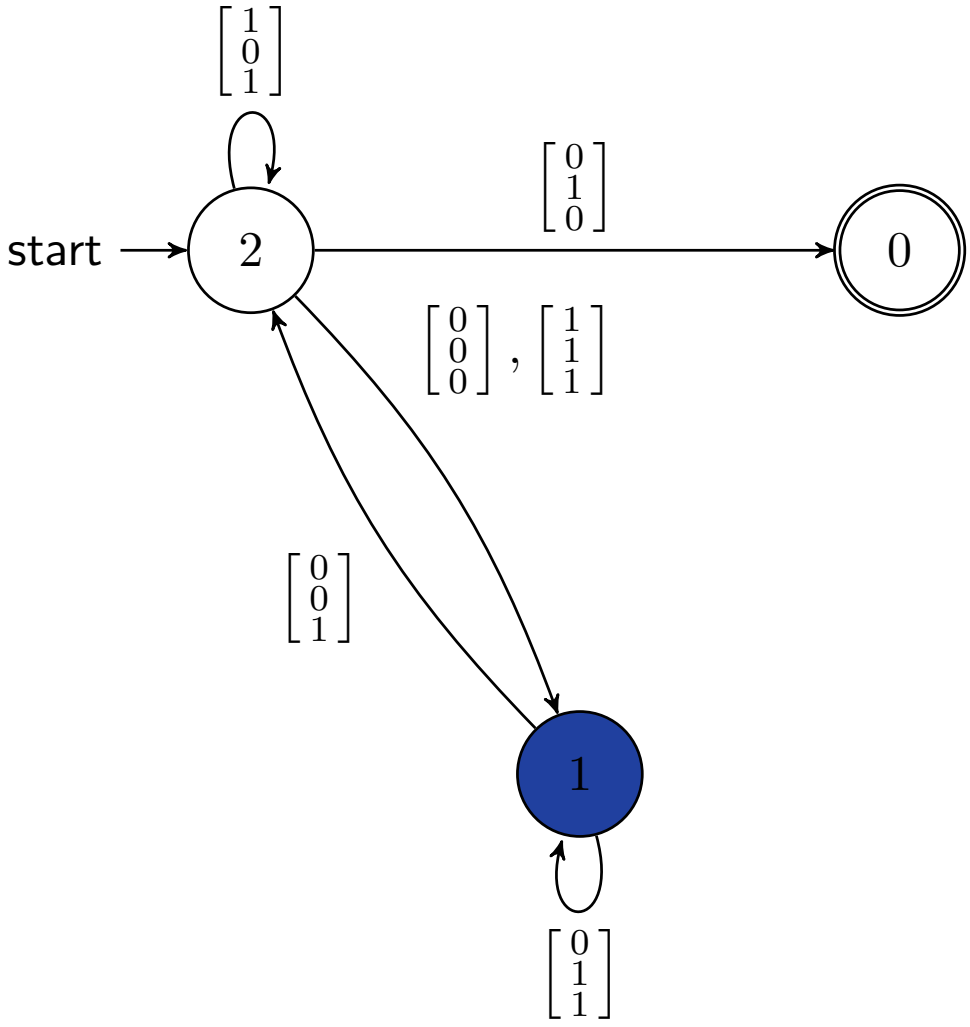

$\begin{bmatrix}1\\0\\1\end{bmatrix}$
start
2
$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
1
$\varphi ::= x + 2y - 3z = 2$



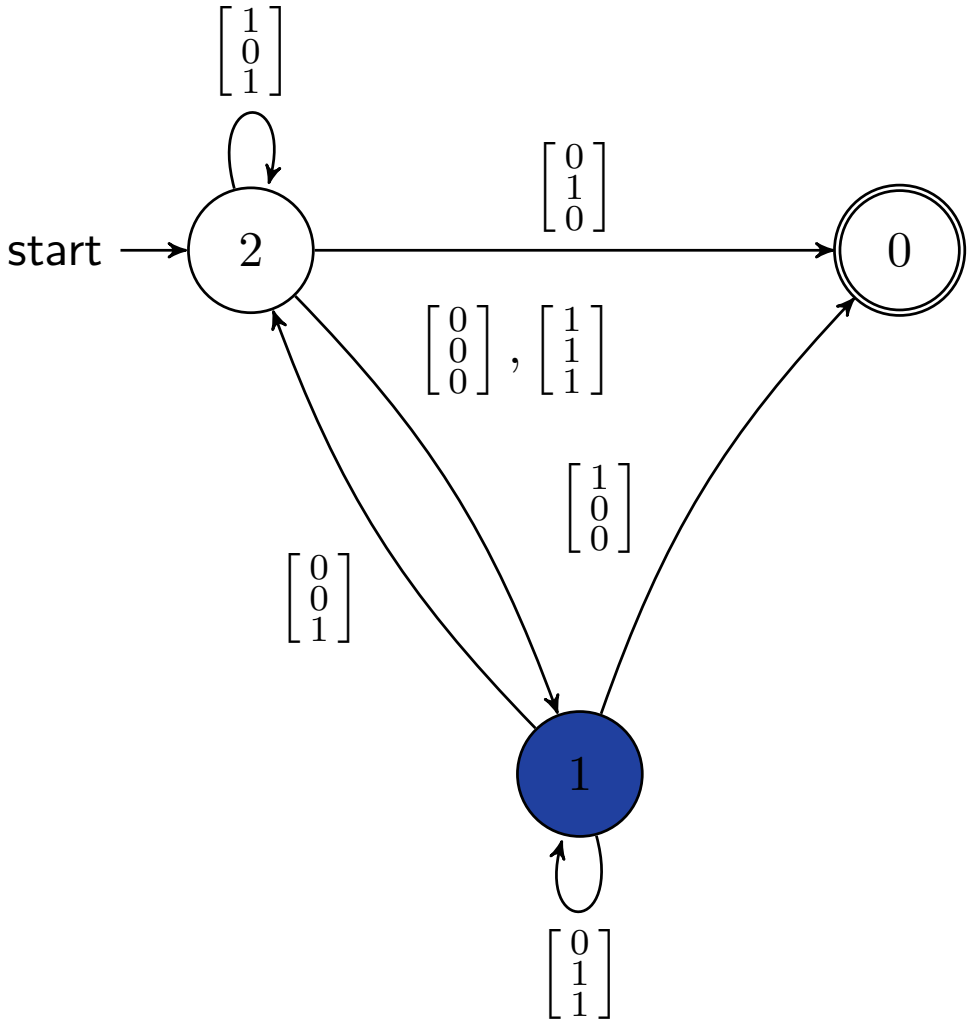
$$\varphi ::= x + 2y - 3z = 2$$



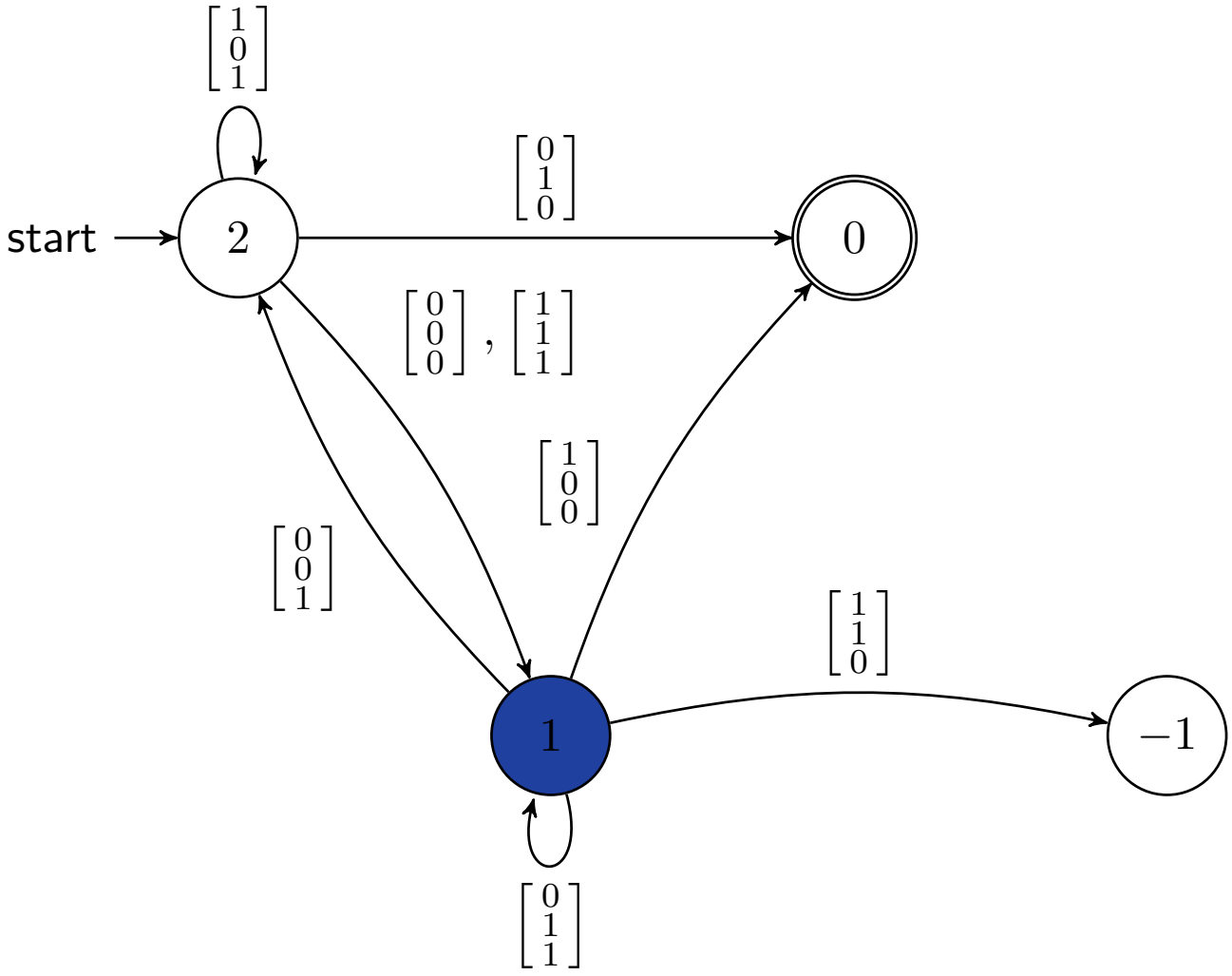
$$\varphi ::= x + 2y - 3z = 2$$



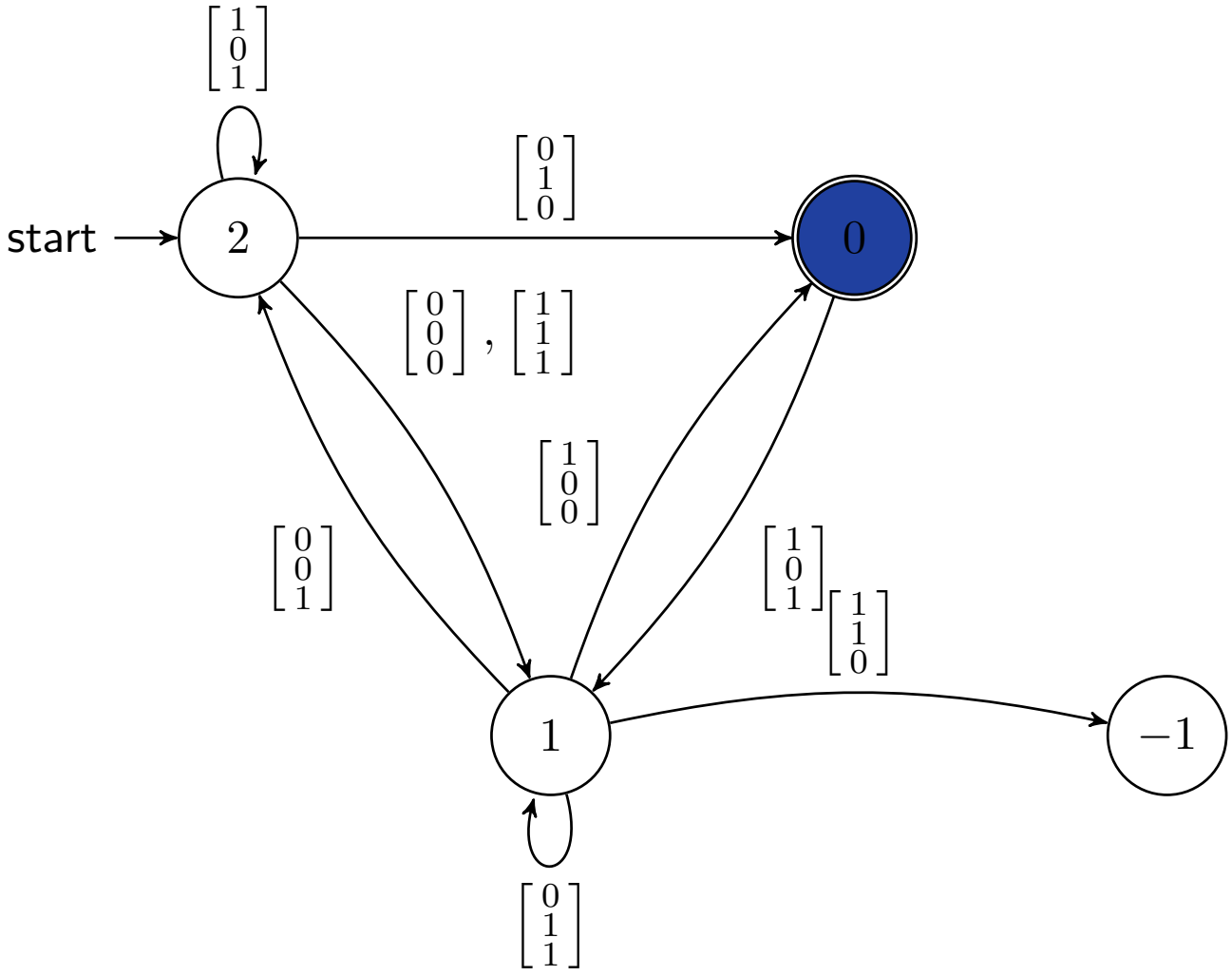
$$\varphi ::= x + 2y - 3z = 2$$



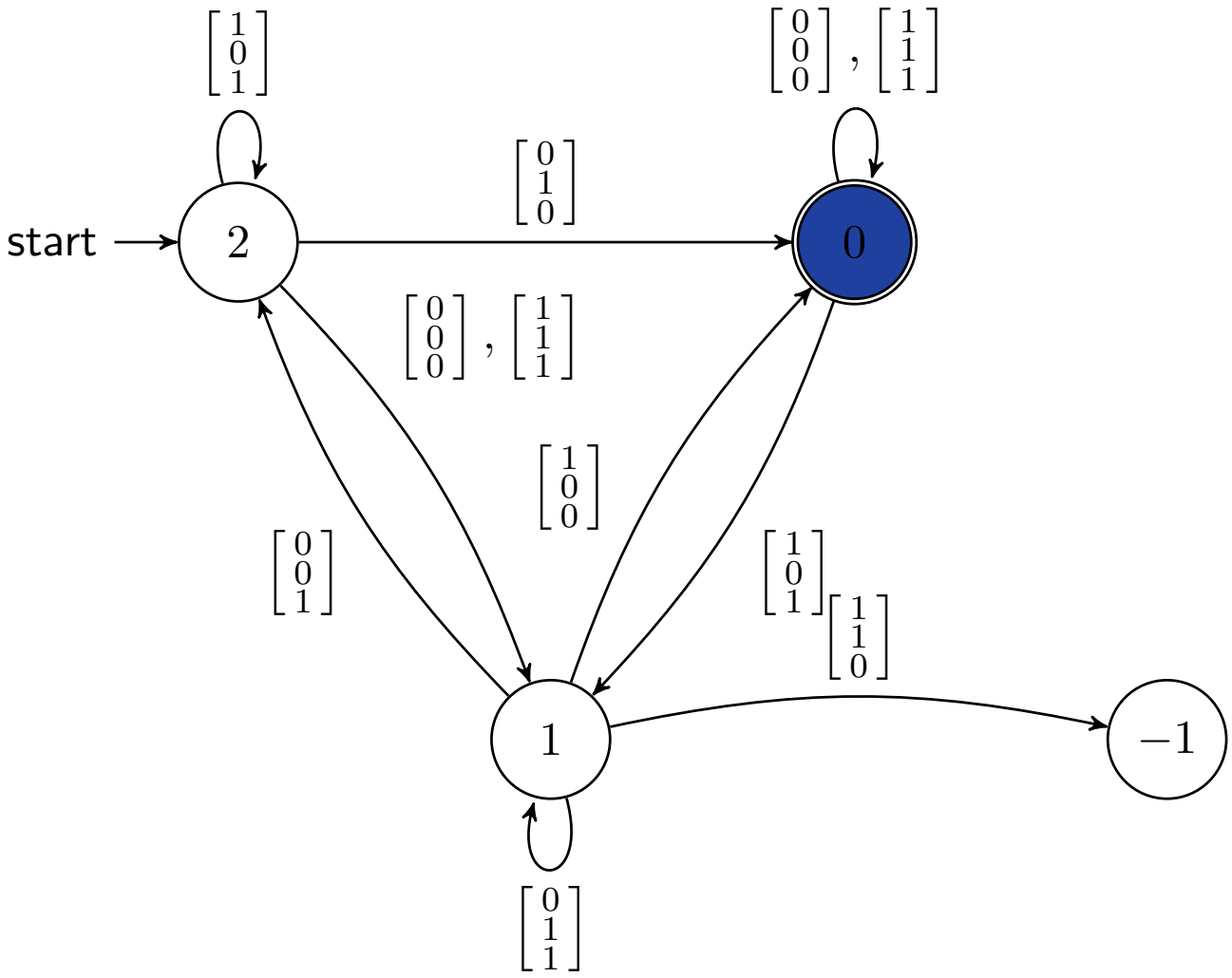
$$\varphi ::= x + 2y - 3z = 2$$



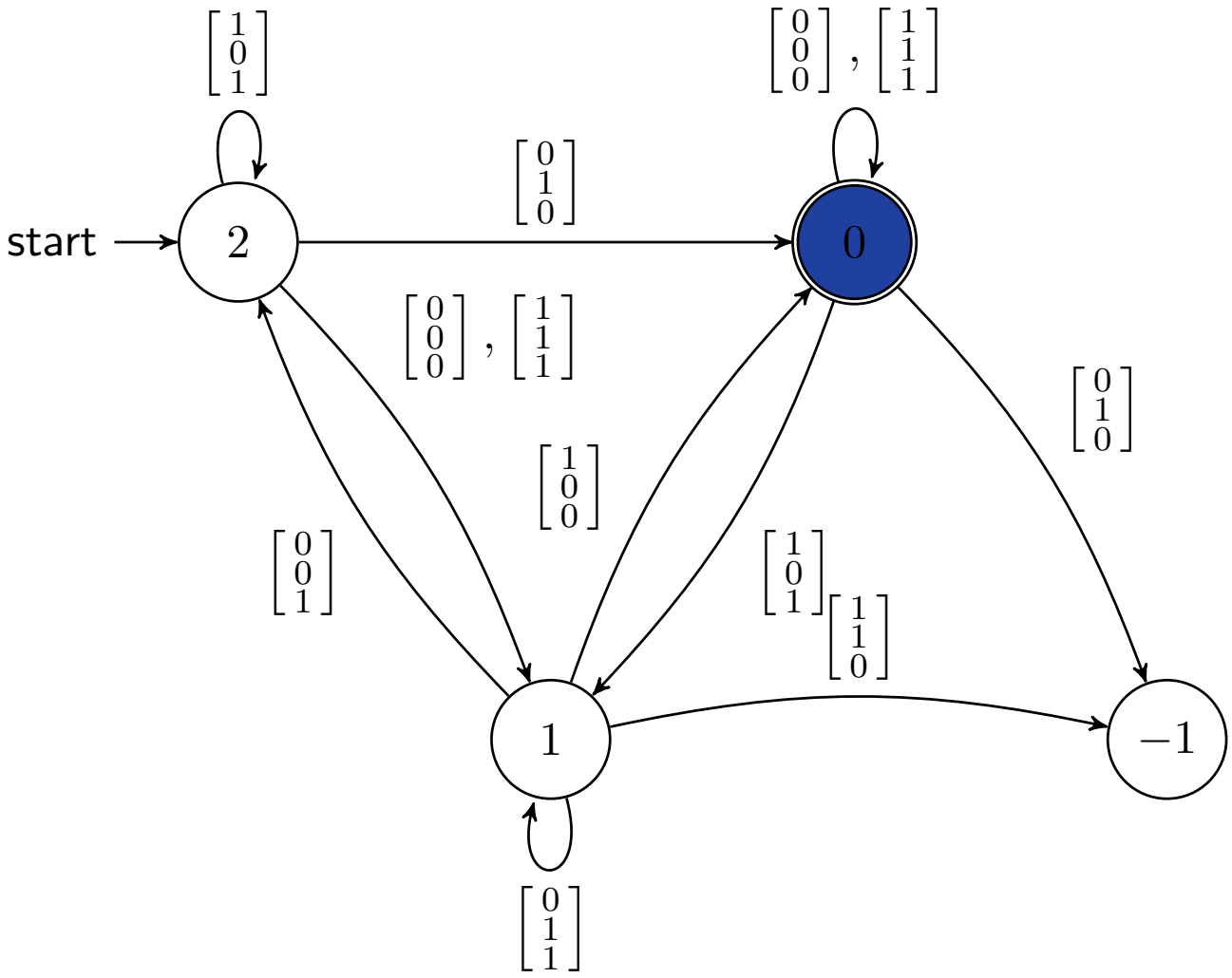
$$\varphi ::= x + 2y - 3z = 2$$



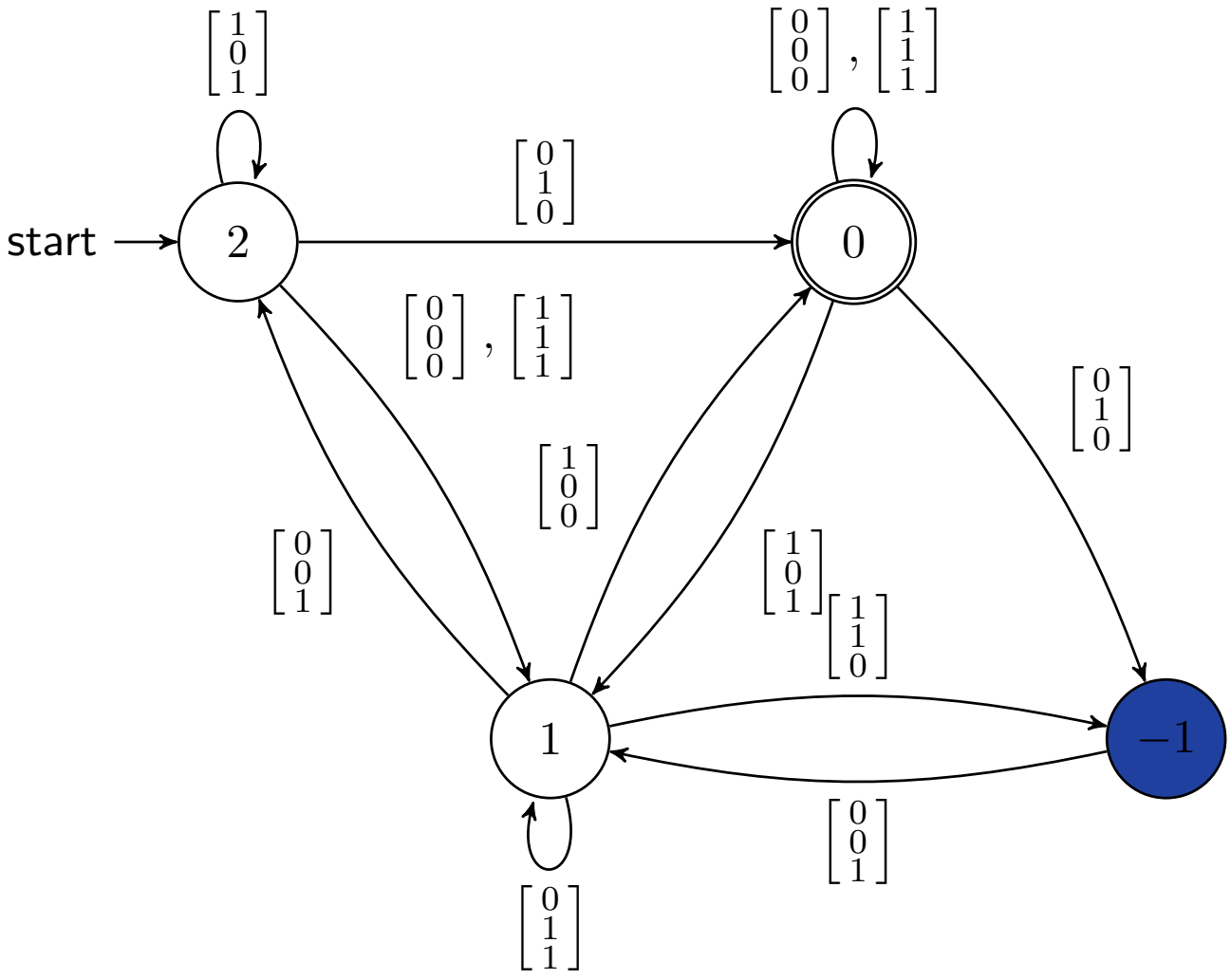
$$\varphi ::= x + 2y - 3z = 2$$



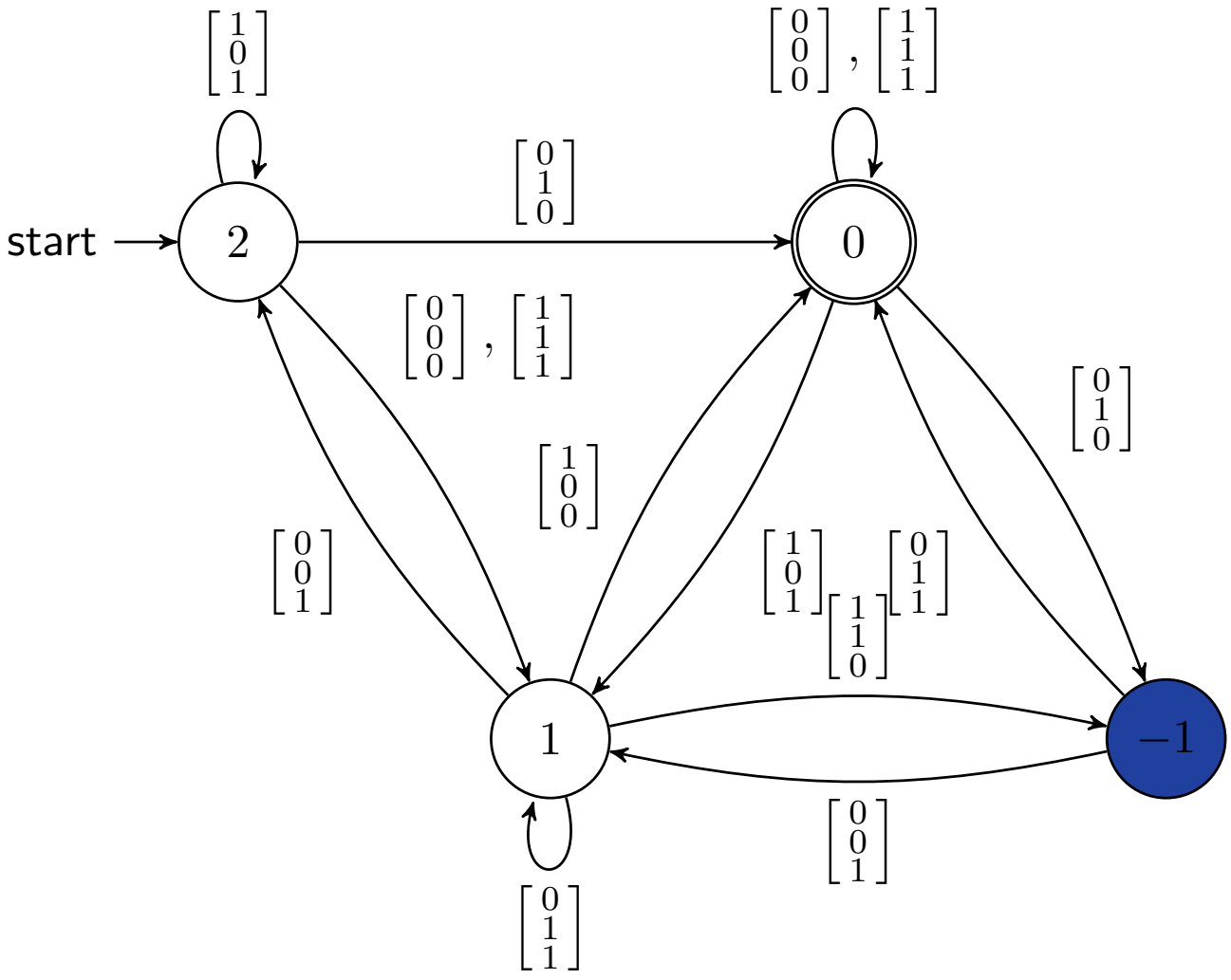
$$\varphi ::= x + 2y - 3z = 2$$



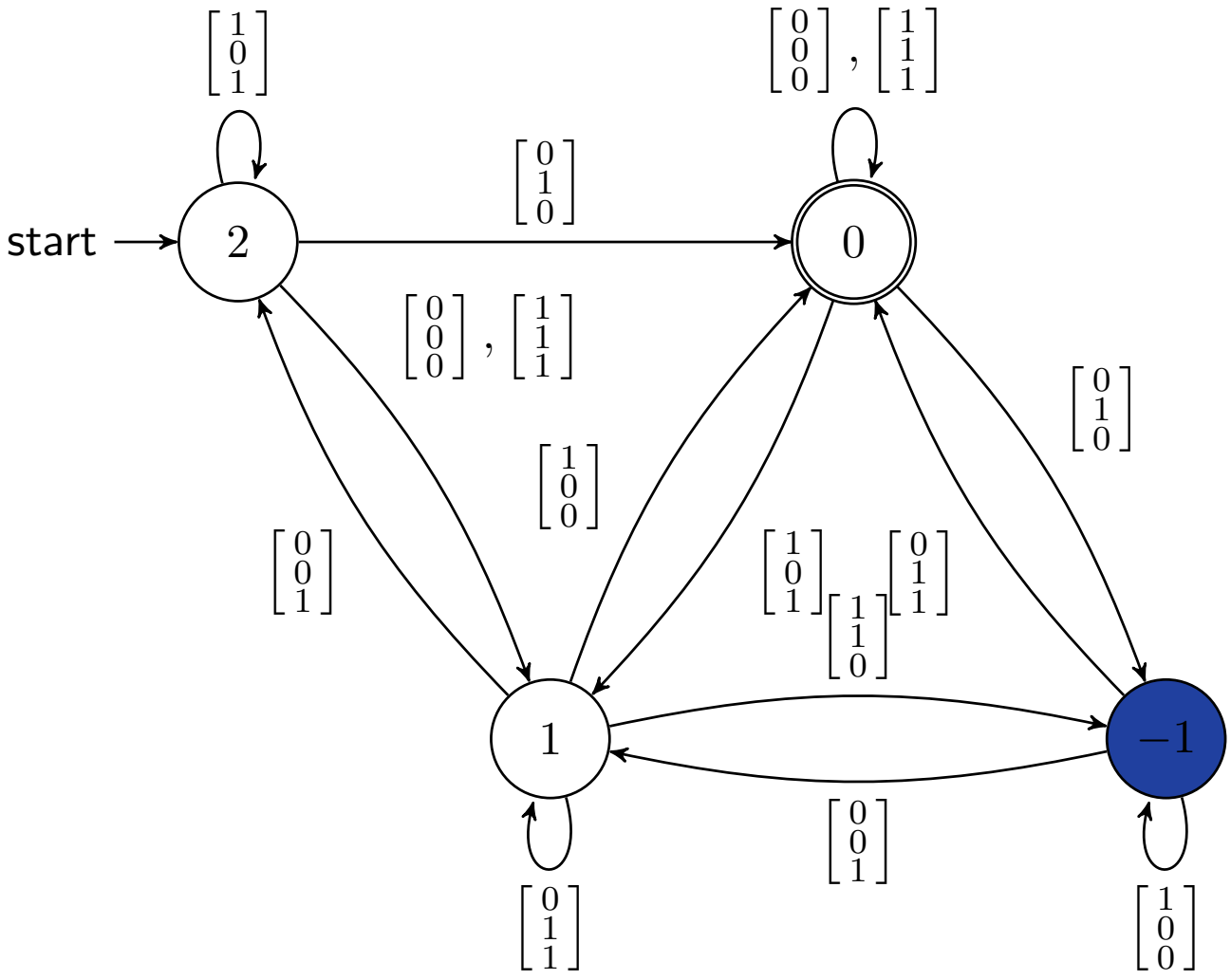
$$\varphi ::= x + 2y - 3z = 2$$



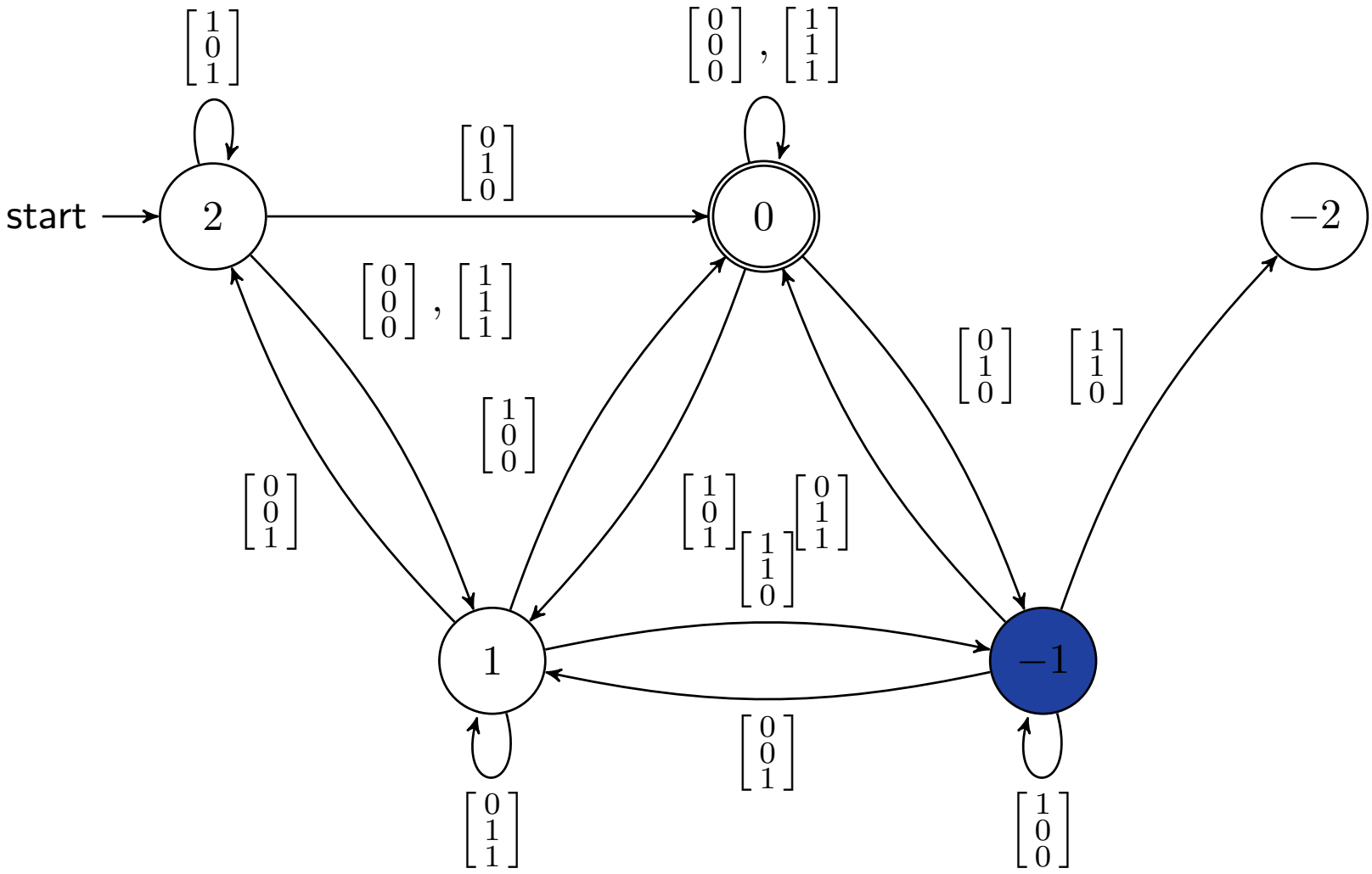
$$\varphi ::= x + 2y - 3z = 2$$



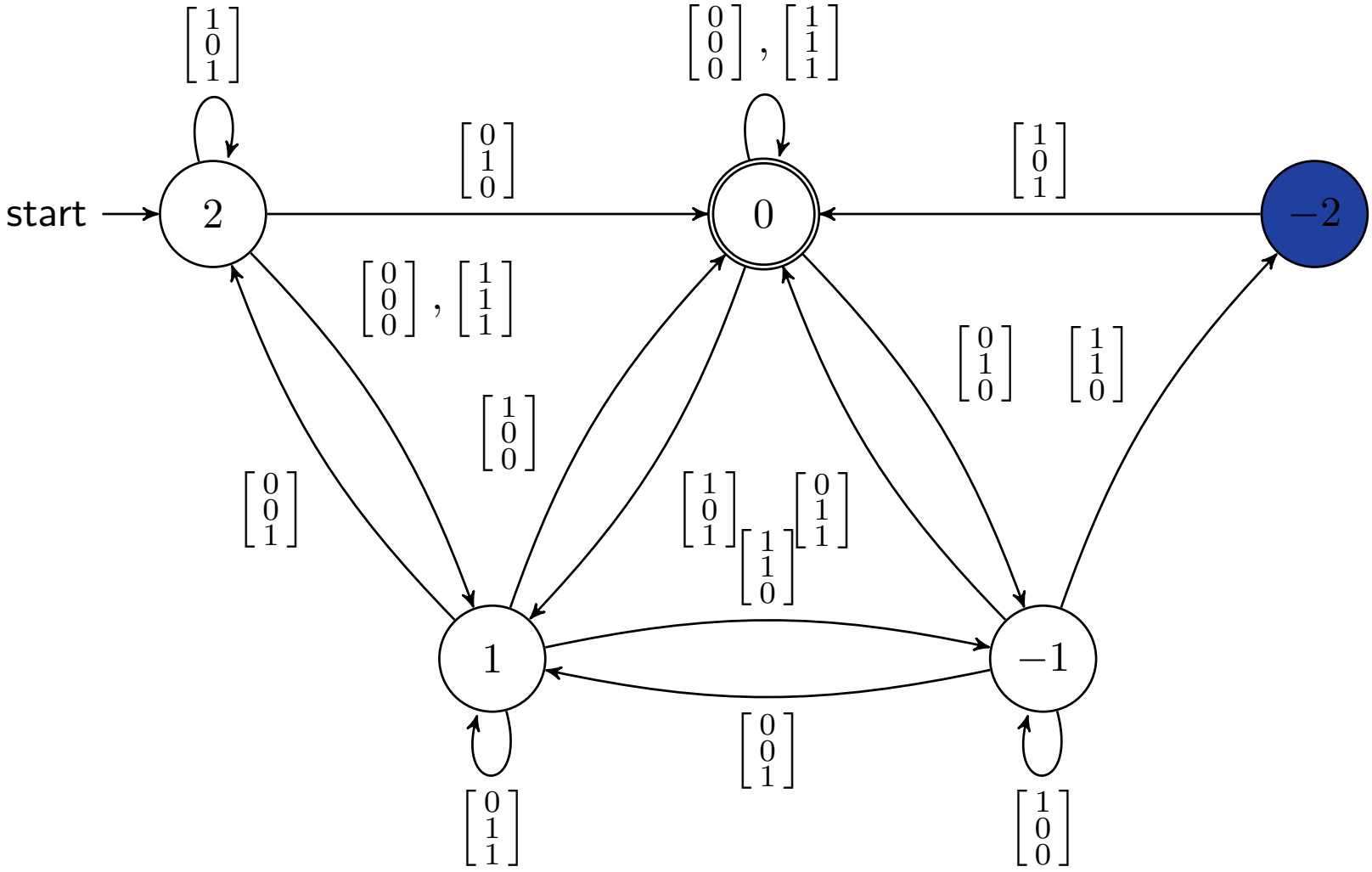
$$\varphi ::= x + 2y - 3z = 2$$



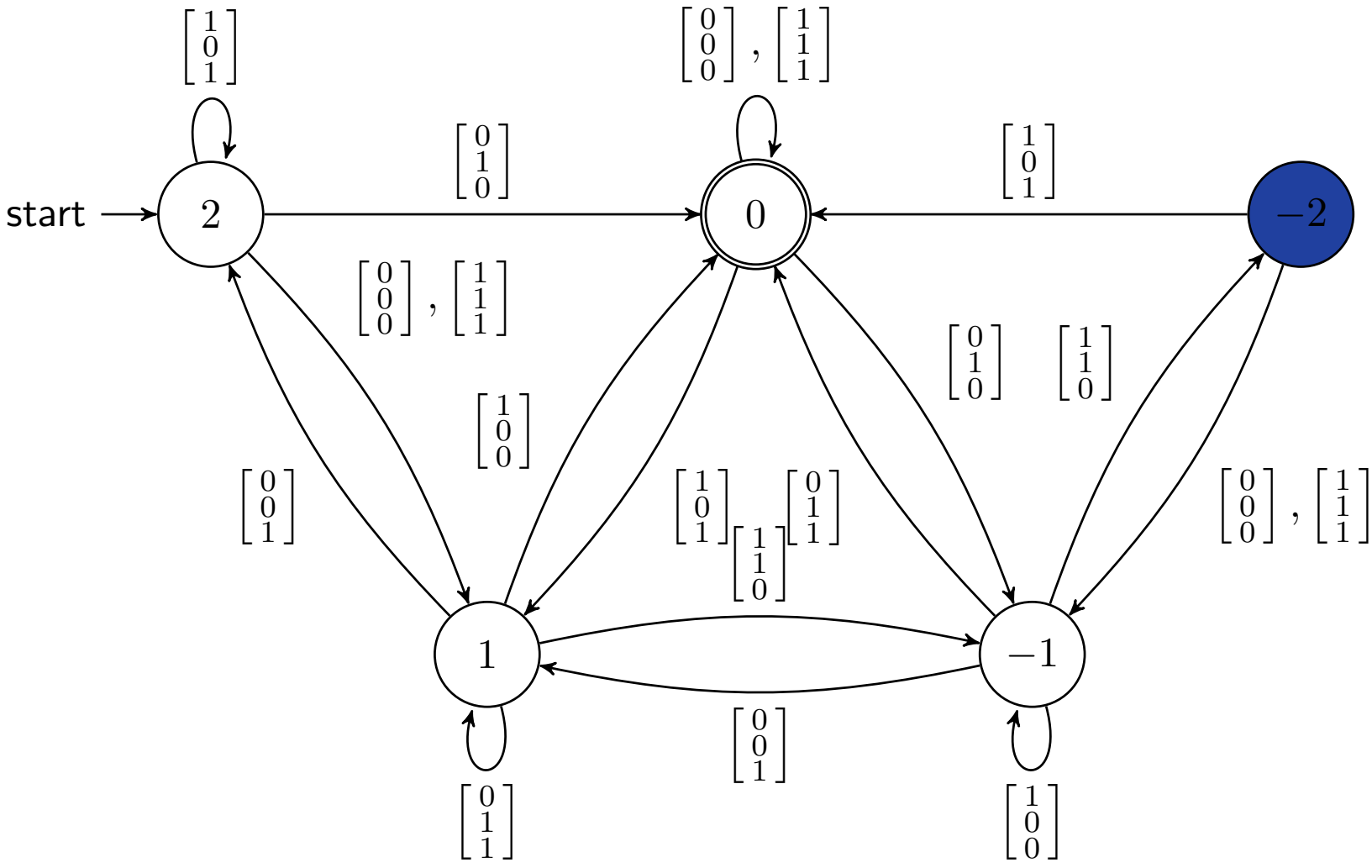
$$\varphi ::= x + 2y - 3z = 2$$



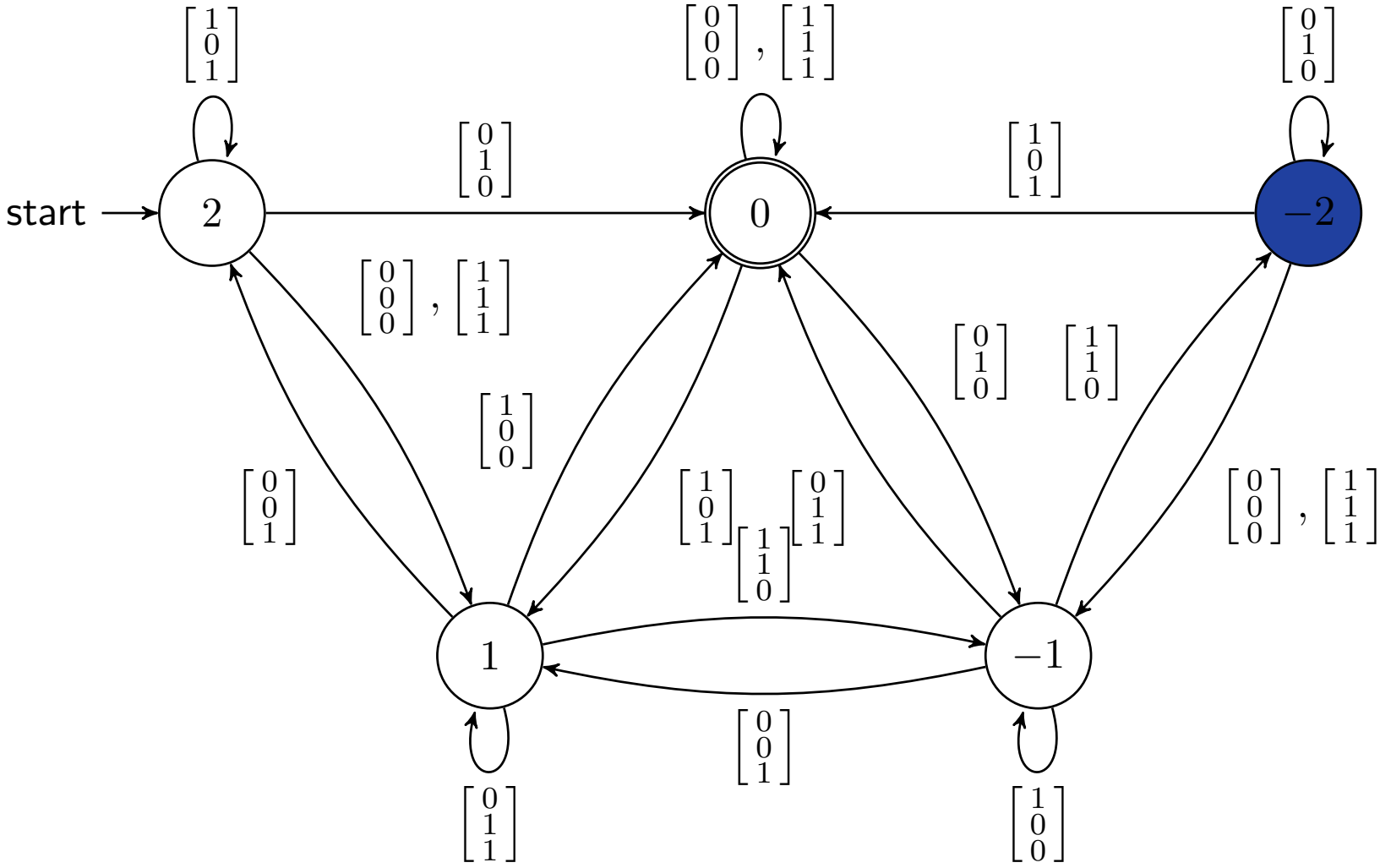
$$\varphi ::= x + 2y - 3z = 2$$



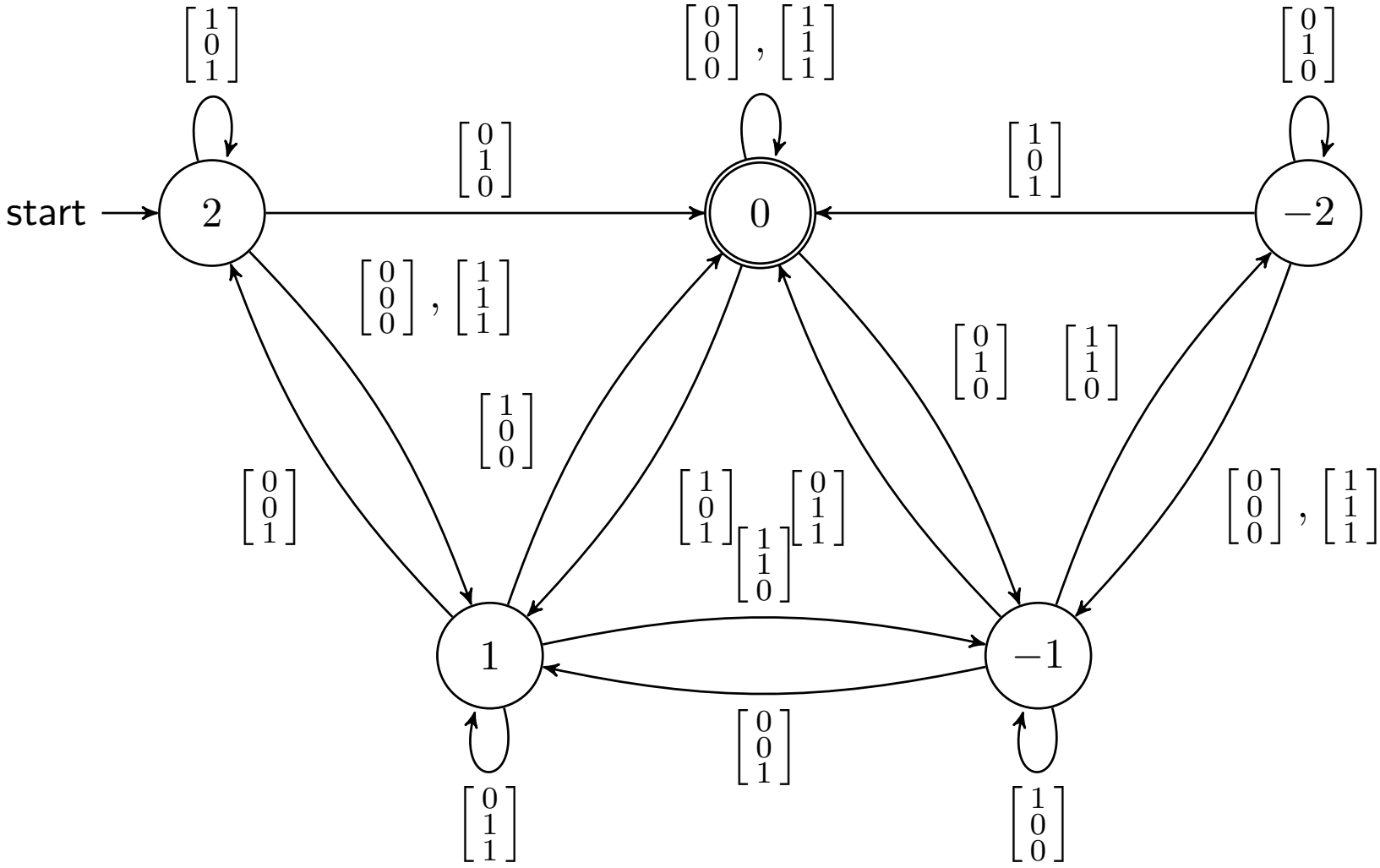
$$\varphi ::= x + 2y - 3z = 2$$



$$\varphi ::= x + 2y - 3z = 2$$



$$\varphi ::= x + 2y - 3z = 2$$



$$\varphi ::= x + 2y - 3z = 2$$

Lemma 1.
Sei $\varphi ::= ax = b$ *und* $s := \sum_{i=1}^{k} |a_i|$ *Für jeden durch* `EqtoDFA`(φ) *erzeugten Zustand* j *gilt,* $-|b| - s \leq j \leq |b| + s$.

Beweis.
Induktion über die Anzahl der erzeugten Zustände.
Basisfall: Ist für s_b erfüllt.
Schritt: Der neue Zustand s_j wird durch s_k erreicht.

$$-|b| - s \leq \frac{-|b| - s - a\zeta}{2} \leq j \leq \frac{|b| + s - a\zeta}{2} \leq \frac{|b| + 2s}{2} \leq |b| + s$$

□

Korollar 1.
`EqtoDFA`(φ) *terminert für jedes* φ.

Lemma 2.
Sei $\varphi ::= ax = b$ *und* A_φ *der von* $\texttt{EqtoDFA}(\varphi)$ *erzeugte Automat.*
$\forall q \in Q. \forall w \in (\{0,1\}^n)^*. \big(w \in L(q) \iff a\texttt{LSBF}^{-1}(w) = q\big)$

Beweis.
Induktion über die Anzahl der erzeugten Zustände.
Basisfall: Defintion der Endzustände.
Schritt: IH + Konstruktionseigenschaft. □

Korollar 2.
$\texttt{EqtoDFA}(\varphi)$ *ist total korrekt.*

1 Grundlagen

2 Automatenkonstruktion
- Kodierung
- Formel
- Gleichungen
- Ungleichungen

3 Ganze Zahlen

$$
\begin{aligned}
\langle \mathit{formula} \rangle &::= \text{`}\neg\text{'}\ \langle \mathit{formula} \rangle \\
&\mid \langle \mathit{formula} \rangle\ \text{`}\vee\text{'}\ \langle \mathit{formula} \rangle \\
&\mid \text{`}\exists\text{'}\ \langle \mathit{var} \rangle\ \langle \mathit{formula} \rangle \\
&\mid \langle \mathit{atomicformula} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle \mathit{atomicformula} \rangle &::= \langle \mathit{term} \rangle\ \text{`}=\text{'}\ \langle \mathit{term} \rangle \\
&\mid \langle \mathit{term} \rangle\ \text{`}\leq\text{'}\ \langle \mathit{term} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle \mathit{term} \rangle &::= \langle \mathit{term} \rangle\ \text{`}+\text{'}\ \langle \mathit{term} \rangle \\
&\mid \langle \mathit{variable} \rangle \\
&\mid \langle \mathit{constant} \rangle
\end{aligned}
$$

$$
\langle \mathit{variable} \rangle ::= \mathrm{x} \mid \mathrm{y} \mid \mathrm{z} \mid \ldots
$$

$$
\langle \mathit{constant} \rangle ::= 0 \mid 1
$$

`IqtoDFA(`φ`)`

Data : Inequation $\varphi ::= ax \leq b$

Result : DFA $A_\varphi = (Q, \Sigma, \delta, q_0, F)$ such that $L(A_\varphi) = L(\varphi)$

$Q, \delta, F \leftarrow \emptyset$;

$q_0 \leftarrow s_b$;

$W \leftarrow \{s_b\}$;

while $W \neq \emptyset$ **do**

- $s_k \leftarrow$ `pick(`W`)`;
- $Q \leftarrow Q \cup \{s_k\}$;
- **if** $k \geq 0$ **then** $F \leftarrow F \cup \{s_k\}$;
- **forall** $\zeta \in \{0,1\}^n$ **do**
  - $j \leftarrow \left\lfloor \frac{1}{2}(k - a\zeta) \right\rfloor$;
  - **if** $s_j \notin Q$ **then** $W \leftarrow W \cup \{s_j\}$;
  - $\delta \leftarrow \delta \cup (s_k, \zeta, s_j)$;

1 Grundlagen

2 Automatenkonstruktion

3 Ganze Zahlen
- Kodierung
- Gleichungen
- Ungleichungen

1 Grundlagen

2 Automatenkonstruktion

3 Ganze Zahlen
- Kodierung
- Gleichungen
- Ungleichungen

- $\Sigma = \{0, 1\}^n$, for free variables $x_1, \ldots x_n$.

- $\Sigma = \{0,1\}^n$, for free variables $x_1, \ldots x_n$.
- $L(\varphi) = \bigcup_{s \in Sol(\varphi)} \texttt{LSBF2}(s)$

- $\Sigma = \{0,1\}^n$, for free variables $x_1, \ldots x_n$.
- $L(\varphi) = \bigcup_{s \in Sol(\varphi)} \texttt{LSBF2}(s)$

Example:

$$\texttt{LSBF2}(4,-3,-2) = \begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}\left(\begin{bmatrix}0\\1\\1\end{bmatrix}\right)^*$$

$$\begin{aligned} 2^2 &= 4 \\ 2^0 + 2^2 - 8 &= -3 \\ 2^1 - 4 &= -2 \end{aligned}$$

1 Grundlagen

2 Automatenkonstruktion

3 Ganze Zahlen
- Kodierung
- Gleichungen
- Ungleichungen

`EqZtoNFA(`φ`)`

Data : Equation $\varphi ::= ax = b$

Result : DFA $A_\varphi = (Q, \Sigma, \delta, q_0, F)$ such that $L(A_\varphi) = L(\varphi)$

$Q, \delta, F \leftarrow \emptyset$;

$q_0 \leftarrow s_b$;

$W \leftarrow \{s_b\}$;

while $W \neq \emptyset$ **do**

  $s_k \leftarrow$ `pick(`W`)`;

  $Q \leftarrow Q \cup \{s_k\}$;

  forall $\zeta \in \{0,1\}^n$ **do**

    if $k - a\zeta$ *is even* **then**

      if $k + a\zeta = 0$ **then** $F \leftarrow F \cup \{s_k\}$;

      $j \leftarrow \frac{1}{2}(k - a\zeta)$;

      if $s_j \notin Q$ **then** $W \leftarrow W \cup \{s_j\}$;

      $\delta \leftarrow \delta \cup (s_k, \zeta, s_j)$;

      $j' \leftarrow k + a\zeta$;

      if $j' \geq 0$ **then**

        $Q \leftarrow Q \cup \{q_f\}$;

        $F \leftarrow F \cup \{q_f\}$;

        $\delta \leftarrow \delta \cup (s_k, \zeta, q_f)$;

1 Grundlagen

2 Automatenkonstruktion

3 Ganze Zahlen
- Kodierung
- Gleichungen
- Ungleichungen

`IqZtoNFA(`φ`)`

Data : Inequation $\varphi ::= ax \leq b$

Result : DFA $A_\varphi = (Q, \Sigma, \delta, q_0, F)$ such that $L(A_\varphi) = L(\varphi)$

$Q, \delta, F \leftarrow \emptyset$;

$q_0 \leftarrow s_b$;

$W \leftarrow \{s_b\}$;

while $W \neq \emptyset$ **do**

  $s_k \leftarrow$ `pick(`W`)`;

  $Q \leftarrow Q \cup \{s_k\}$;

  forall $\zeta \in \{0,1\}^n$ **do**

    $j \leftarrow \left\lfloor \frac{1}{2}(k - a\zeta) \right\rfloor$;

    if $s_j \notin Q$ **then** $W \leftarrow W \cup \{s_j\}$;

    $\delta \leftarrow \delta \cup (s_k, \zeta, s_j)$;

    $j' \leftarrow k + a\zeta$;

    if $j' \geq 0$ **then**

      $Q \leftarrow Q \cup \{q_f\}$;

      $F \leftarrow F \cup \{q_f\}$;

      $\delta \leftarrow \delta \cup (s_k, \zeta, q_f)$;