GO ist PSPACE-schwer

Beitrag zum Seminar über Algorithmen und Komplexität
Freie Universität Berlin, SS 07

Frederik Hermans

24. Mai 2007

Überblick

- GO ist ein asiatisches Brettspiel.
- Lichtenstein und Sipser haben bewiesen, dass GO PSPACE-schwer ist.
- Der Beweis geht wie folgt:

Überblick

- GO ist ein asiatisches Brettspiel.
- Lichtenstein und Sipser haben bewiesen, dass GO PSPACE-schwer ist.
- Der Beweis geht wie folgt:
  1. Betrachte zunächst das Problem TQBF.

Überblick

- GO ist ein asiatisches Brettspiel.
- Lichtenstein und Sipser haben bewiesen, dass GO PSPACE-schwer ist.
- Der Beweis geht wie folgt:
  1. Betrachte zunächst das Problem TQBF.
  2. Reduziere TQBF auf GG.

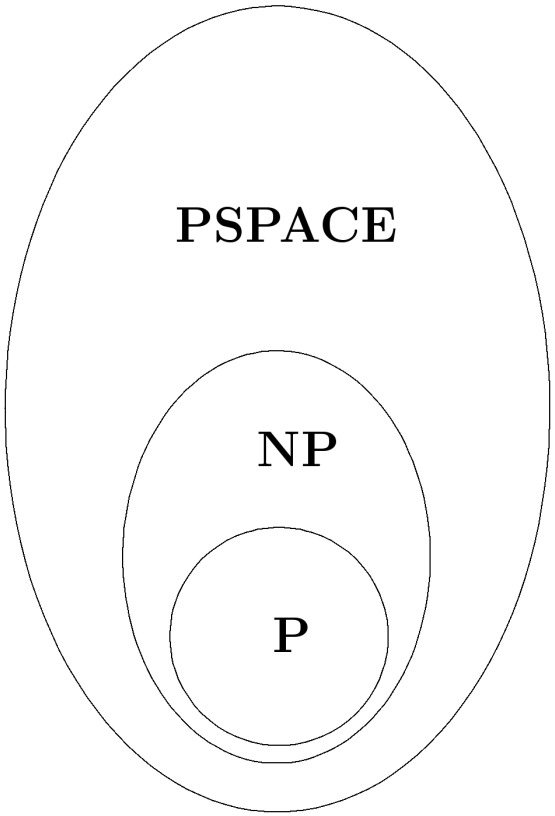
Überblick

- GO ist ein asiatisches Brettspiel.
- Lichtenstein und Sipser haben bewiesen, dass GO PSPACE-schwer ist.
- Der Beweis geht wie folgt:
  1. Betrachte zunächst das Problem TQBF.
  2. Reduziere TQBF auf GG.
  3. Betrachte GG mit planaren Graphen (PGG).

Überblick

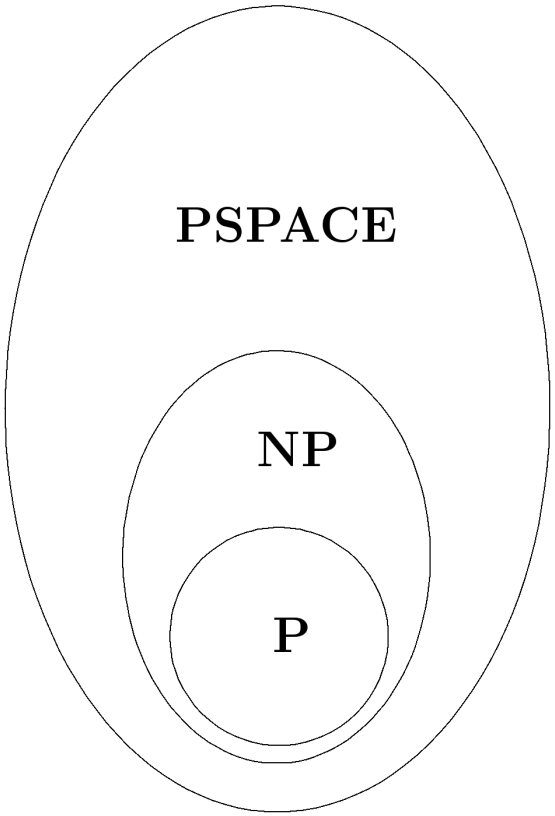
- GO ist ein asiatisches Brettspiel.
- Lichtenstein und Sipser haben bewiesen, dass GO PSPACE-schwer ist.
- Der Beweis geht wie folgt:
  1. Betrachte zunächst das Problem TQBF.
  2. Reduziere TQBF auf GG.
  3. Betrachte GG mit planaren Graphen (PGG).
  4. Reduziere PGG auf GO.

Die Komplexitätsklasse PSPACE



- PSPACE beinhaltet alle Entscheidungsprobleme, deren Lösung auf einer Turing-Maschine polynomiell viel Platz benötigt.
- PSPACE-schwere Probleme gelten gemeinhein als so schwer, dass sie in der Praxis im Allgemeinen nicht lösbar sind.

Die Komplexitätsklasse PSPACE



- PSPACE beinhaltet alle Entscheidungsprobleme, deren Lösung auf einer Turing-Maschine polynomiell viel Platz benötigt.
- PSPACE-schwere Probleme gelten gemeinhein als so schwer, dass sie in der Praxis im Allgemeinen nicht lösbar sind.
- In diesem Beweis wird nur gezeigt, dass GO PSPACE-schwer ist.
- Es ist nicht bekannt, ob GO auch in PSPACE liegt.

Das Problem TQBF
True Quantified Boolean Formulas

- *QBF* ist die Menge der quantifizierten Booleschen Formeln in konjunktiver Normalform.

$$QBF = \{Q_1 v_1 Q_2 v_2 \dots Q_n v_n : F(v_1, v_2, \dots, v_n)\},$$

mit Q_i Quantoren ($\forall$ oder $\exists$), v_i Variablen, $F(v_1, v_2, \dots, v_n)$ eine Boolesche Formel in KNF.

Das Problem TQBF
True Quantified Boolean Formulas

- *QBF* ist die Menge der quantifizierten Booleschen Formeln in konjunktiver Normalform.

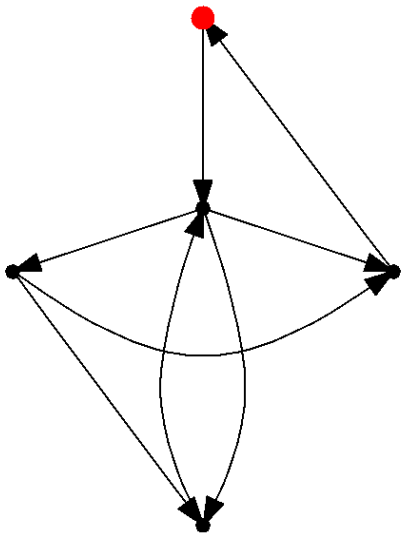
$$QBF = \{Q_1 v_1 Q_2 v_2 \dots Q_n v_n : F(v_1, v_2, \dots, v_n)\},$$

  mit Q_i Quantoren ($\forall$ oder $\exists$), v_i Variablen, $F(v_1, v_2, \dots, v_n)$ eine Boolesche Formel in KNF.
- *TQBF* ist die Menge aller Formeln aus *QBF*, die wahr sind.

Satz (Meyer & Stockmeyer, 1973)

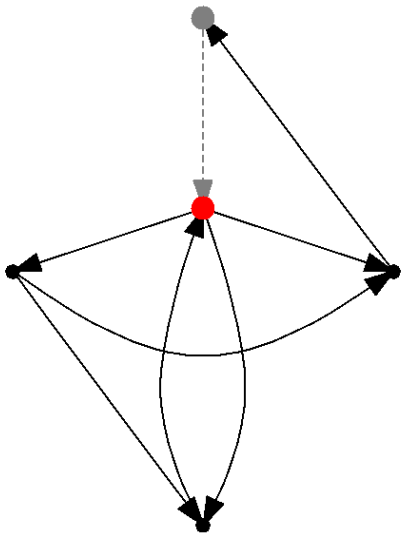
Das Problem, für eine gegebene Formel aus *QBF* zu prüfen, ob sie in *TQBF* liegt, ist PSPACE-vollständig.

Das Problem GG
Definition, Beispiel



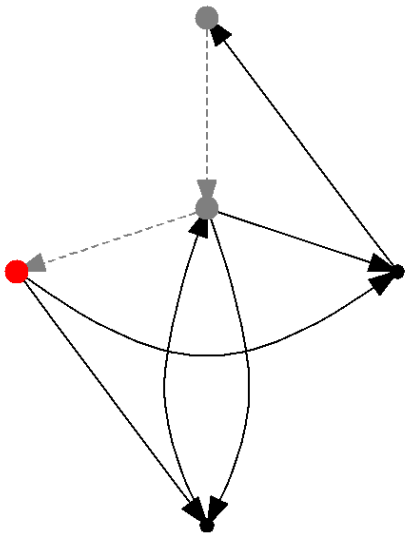
- GG (Generalized Geography) wird von zwei Spielern auf einem gerichteten Graphen gespielt.
- Das Spiel beginnt an einer ausgezeichneten, markierten Startecke.
- Die Spieler wählen abwechselnd von der aktuellen Ecke eine ausgehende Kante, deren Endecke unmarkiert ist. Die Endecke der gewählten Kante wird nun markiert und das Spiel dort fortgesetzt.
- Derjenige Spieler, der zuerst keine Kante mehr auswählen kann, verliert.

Das Problem GG
Definition, Beispiel



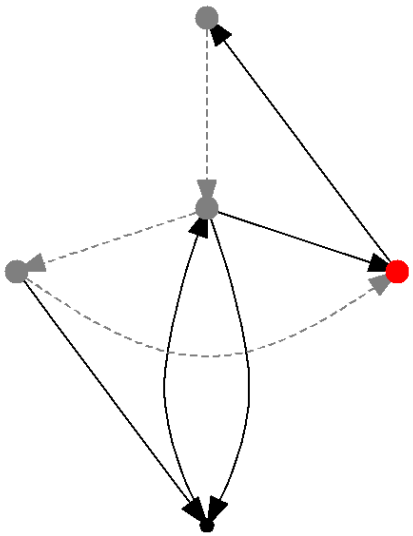
- GG (Generalized Geography) wird von zwei Spielern auf einem gerichteten Graphen gespielt.
- Das Spiel beginnt an einer ausgezeichneten, markierten Startecke.
- Die Spieler wählen abwechselnd von der aktuellen Ecke eine ausgehende Kante, deren Endecke unmarkiert ist. Die Endecke der gewählten Kante wird nun markiert und das Spiel dort fortgesetzt.
- Derjenige Spieler, der zuerst keine Kante mehr auswählen kann, verliert.

Das Problem GG
Definition, Beispiel



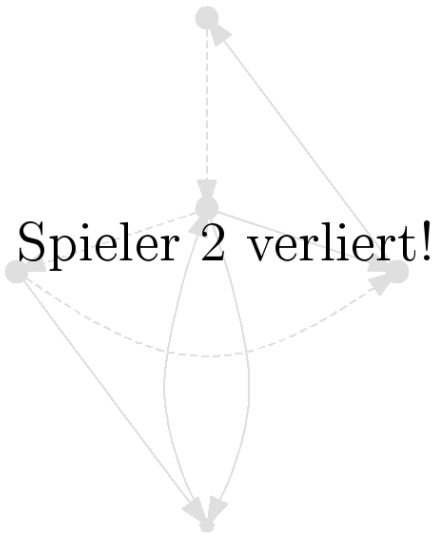
- GG (Generalized Geography) wird von zwei Spielern auf einem gerichteten Graphen gespielt.
- Das Spiel beginnt an einer ausgezeichneten, markierten Startecke.
- Die Spieler wählen abwechselnd von der aktuellen Ecke eine ausgehende Kante, deren Endecke unmarkiert ist. Die Endecke der gewählten Kante wird nun markiert und das Spiel dort fortgesetzt.
- Derjenige Spieler, der zuerst keine Kante mehr auswählen kann, verliert.

Das Problem GG
Definition, Beispiel



- GG (Generalized Geography) wird von zwei Spielern auf einem gerichteten Graphen gespielt.
- Das Spiel beginnt an einer ausgezeichneten, markierten Startecke.
- Die Spieler wählen abwechselnd von der aktuellen Ecke eine ausgehende Kante, deren Endecke unmarkiert ist. Die Endecke der gewählten Kante wird nun markiert und das Spiel dort fortgesetzt.
- Derjenige Spieler, der zuerst keine Kante mehr auswählen kann, verliert.

Das Problem GG
Definition, Beispiel



- GG (Generalized Geography) wird von zwei Spielern auf einem gerichteten Graphen gespielt.
- Das Spiel beginnt an einer ausgezeichneten, markierten Startecke.
- Die Spieler wählen abwechselnd von der aktuellen Ecke eine ausgehende Kante, deren Endecke unmarkiert ist. Die Endecke der gewählten Kante wird nun markiert und das Spiel dort fortgesetzt.
- Derjenige Spieler, der zuerst keine Kante mehr auswählen kann, verliert.

Satz

Das Problem, den Gewinner eines GG-Spiels festzustellen, ist PSPACE-schwer.

Satz

Das Problem, den Gewinner eines GG-Spiels festzustellen, ist PSPACE-schwer.

Beweis:

- Idee: Reduktion TQBF $\leq$ GG.

Satz

Das Problem, den Gewinner eines GG-Spiels festzustellen, ist PSPACE-schwer.

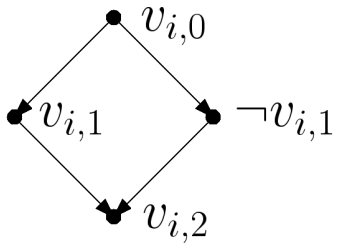
Beweis:

- Idee: Reduktion TQBF $\leq$ GG.
- Gegeben sei eine Formel $B \in QBF$

$$B = \exists v_1 \forall v_2 \exists v_3 \forall v_4 \ldots \exists v_{n-1} \forall v_n : F(v_1, \ldots, v_n).$$

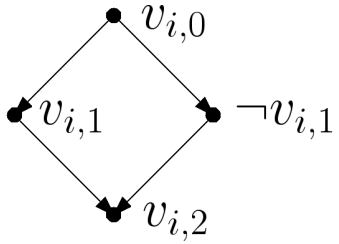
- Konstruiere zu B einen gerichteten Graphen, auf dem GG gespielt wird. Der beginnende Spieler soll dann gewinnen (können), wenn B in $TQBF$ liegt.

- Jede Variable v_i wird durch eine Rhombusstruktur dargestellt:



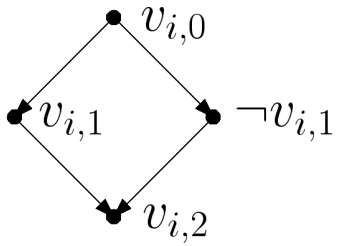
- Jede Klausel j wird durch eine Ecke k_j dargestellt.

- Jede Variable v_i wird durch eine Rhombusstruktur dargestellt:



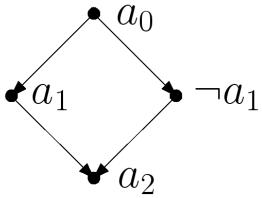
- Jede Klausel j wird durch eine Ecke k_j dargestellt.
- Füge Kanten $(v_{i,2}, v_{i+1,0})$ ein für $1 \leq i < n$.
- Füge Kanten $(v_{n,2}, k_j)$ ein für $1 \leq j \leq m$.
- Füge Wege der Länge 2 ein von k_j nach $v_{i,1}$, wenn v_i in Klausel j, von k_j nach $\neg v_{i,1}$, wenn $\neg v_i$ in Klausel j.

- Jede Variable v_i wird durch eine Rhombusstruktur dargestellt:



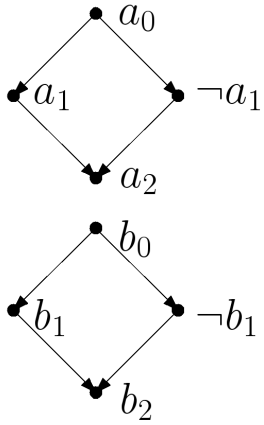
- Jede Klausel j wird durch eine Ecke k_j dargestellt.
- Füge Kanten $(v_{i,2}, v_{i+1,0})$ ein für $1 \leq i < n$.
- Füge Kanten $(v_{n,2}, k_j)$ ein für $1 \leq j \leq m$.
- Füge Wege der Länge 2 ein von k_j nach $v_{i,1}$, wenn v_i in Klausel j, von k_j nach $\neg v_{i,1}$, wenn $\neg v_i$ in Klausel j.
- Die Ecke $v_{1,0}$ ist die Startecke.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



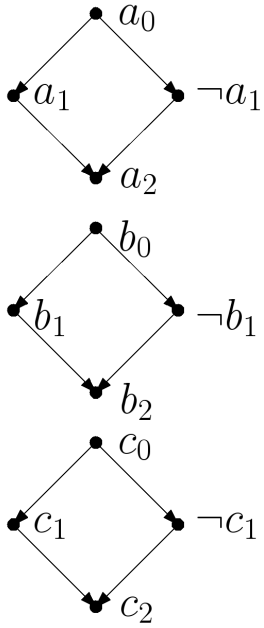
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



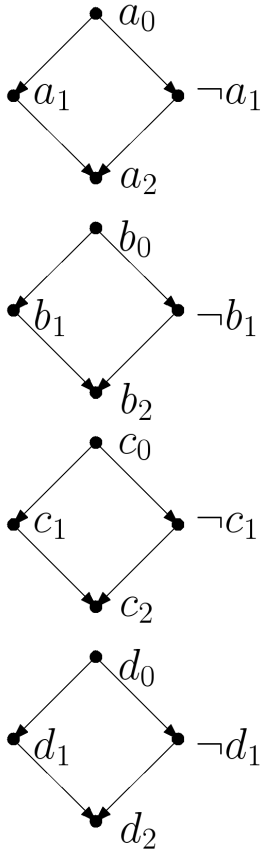
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



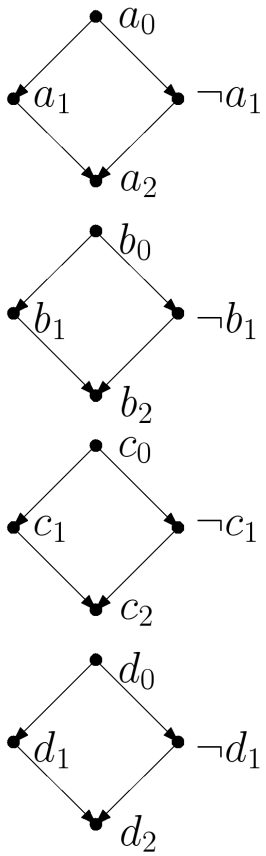
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

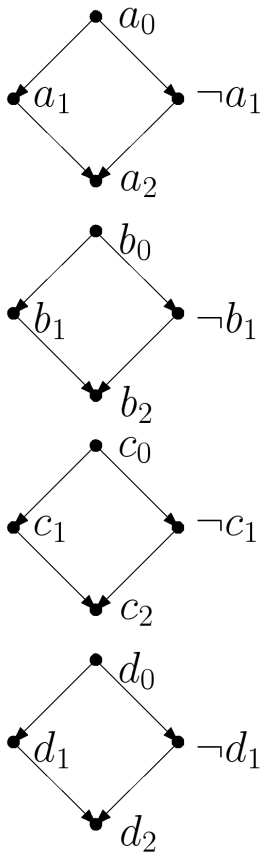
Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

- k_1

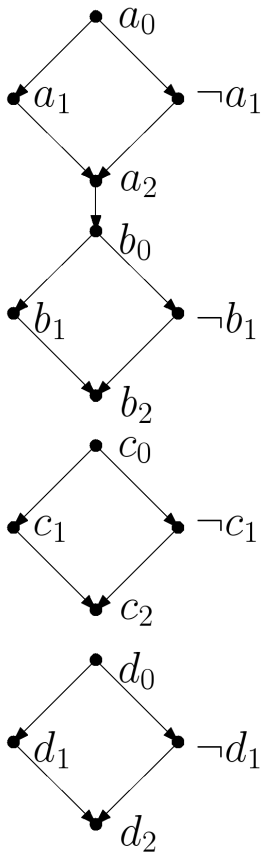
Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

- k_1
- k_2

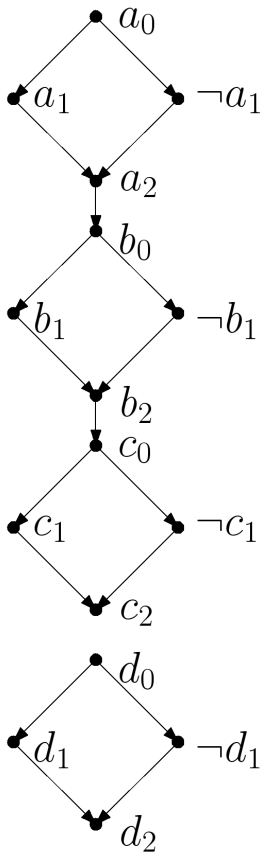
Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

- k_1
- k_2

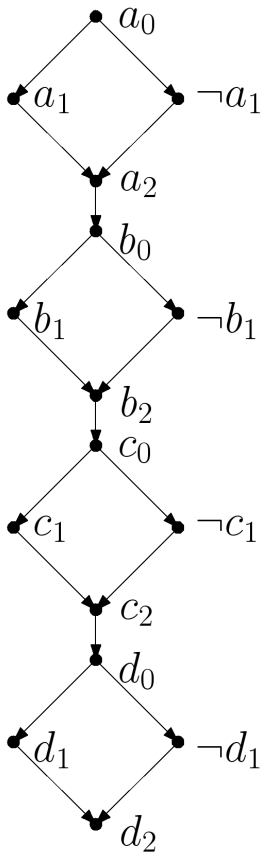
Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

- k_1
- k_2

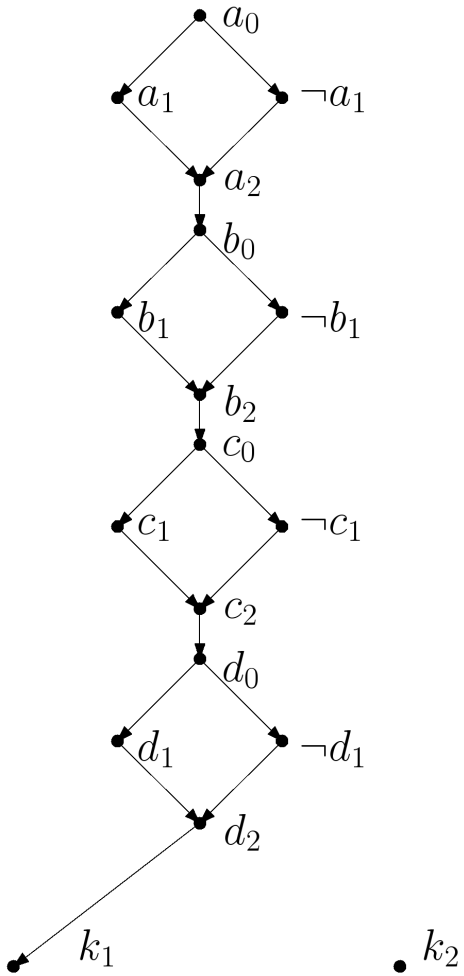
Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

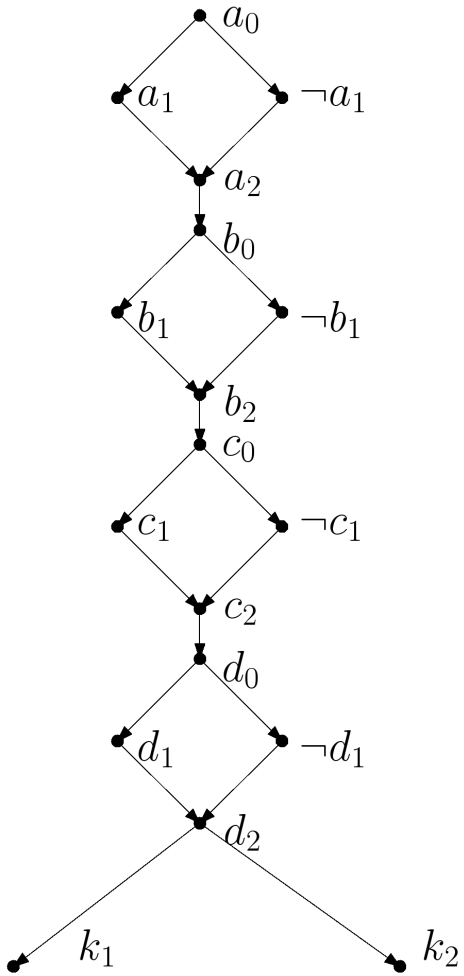
- k_1
- k_2

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



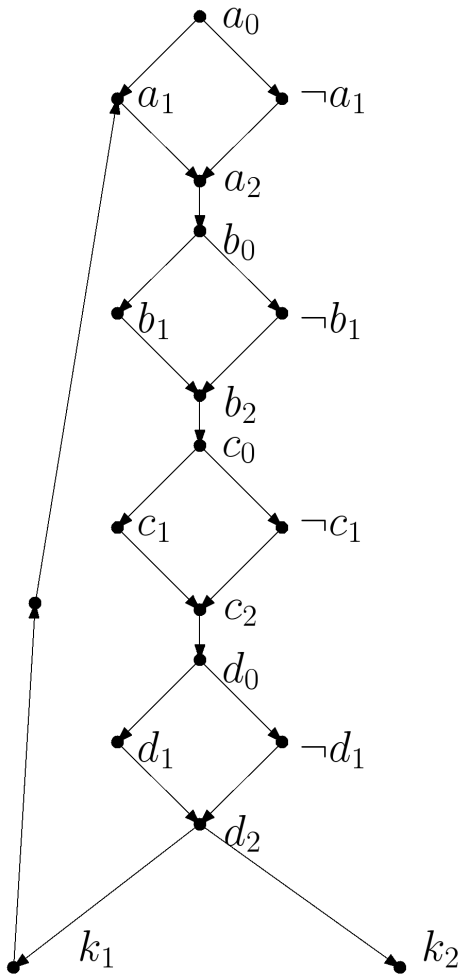
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



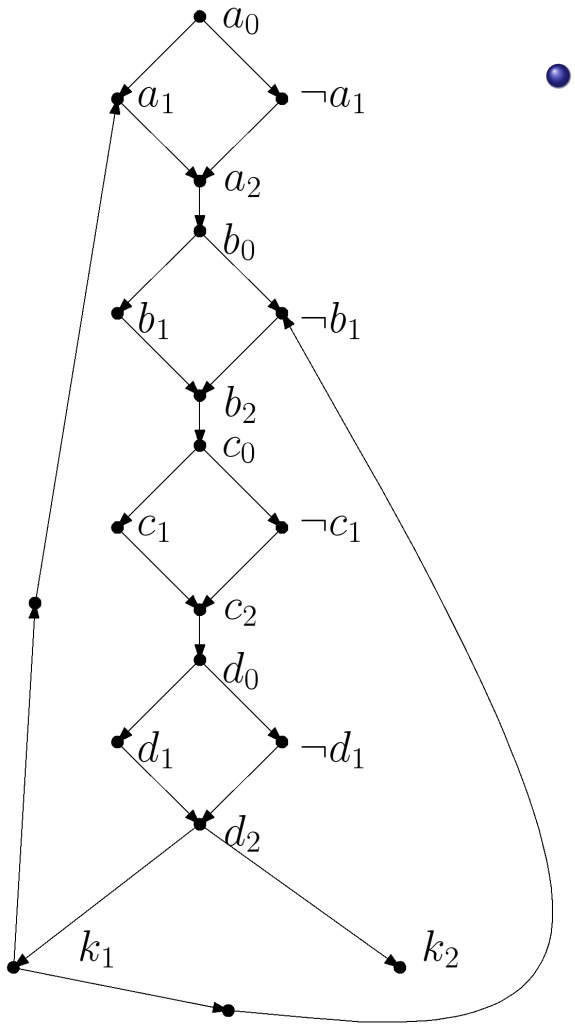
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



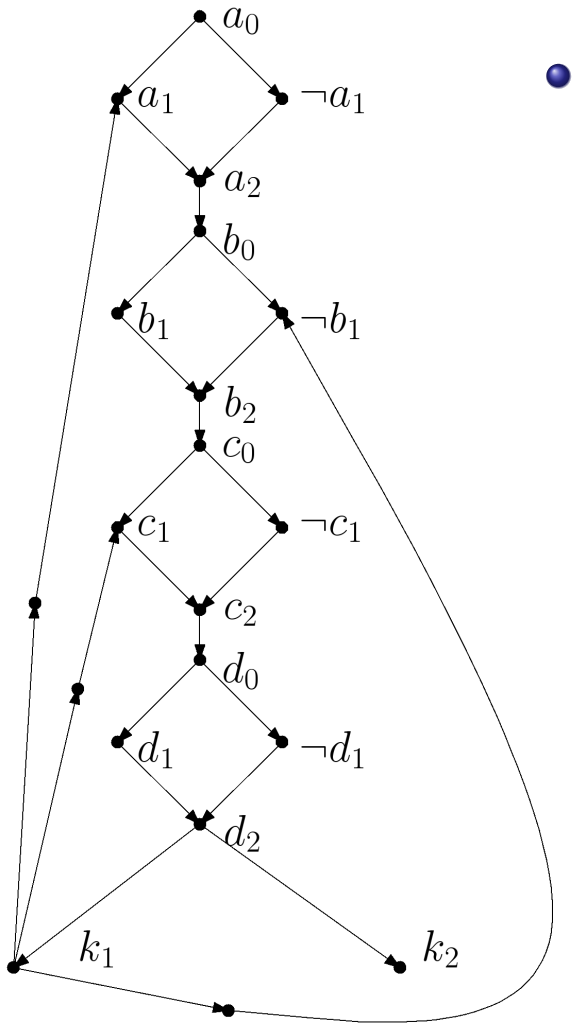
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



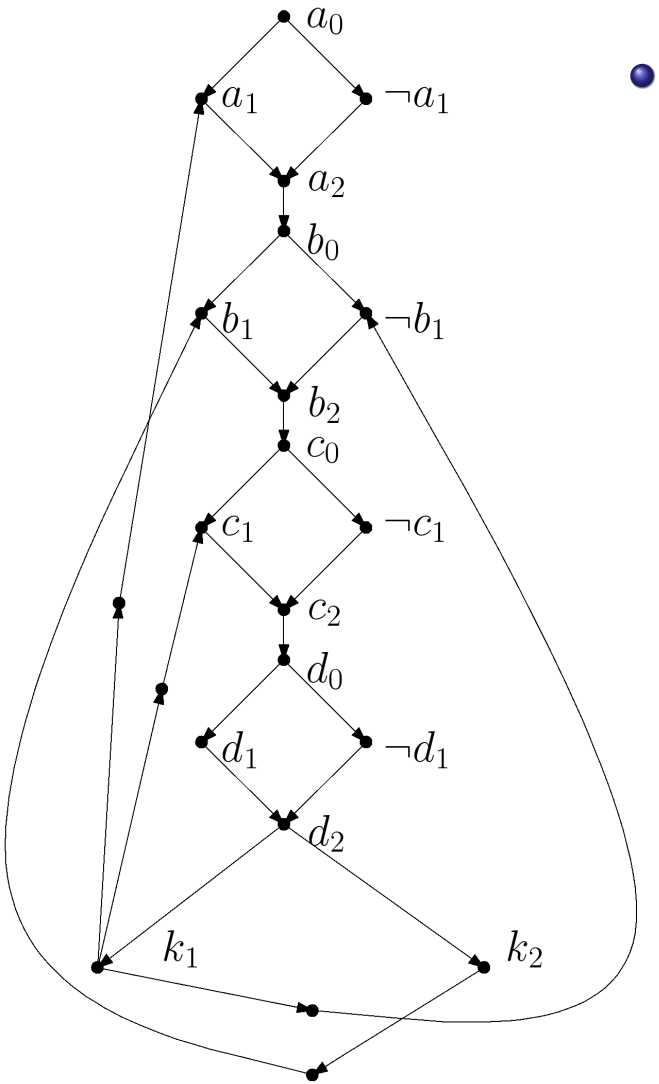
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



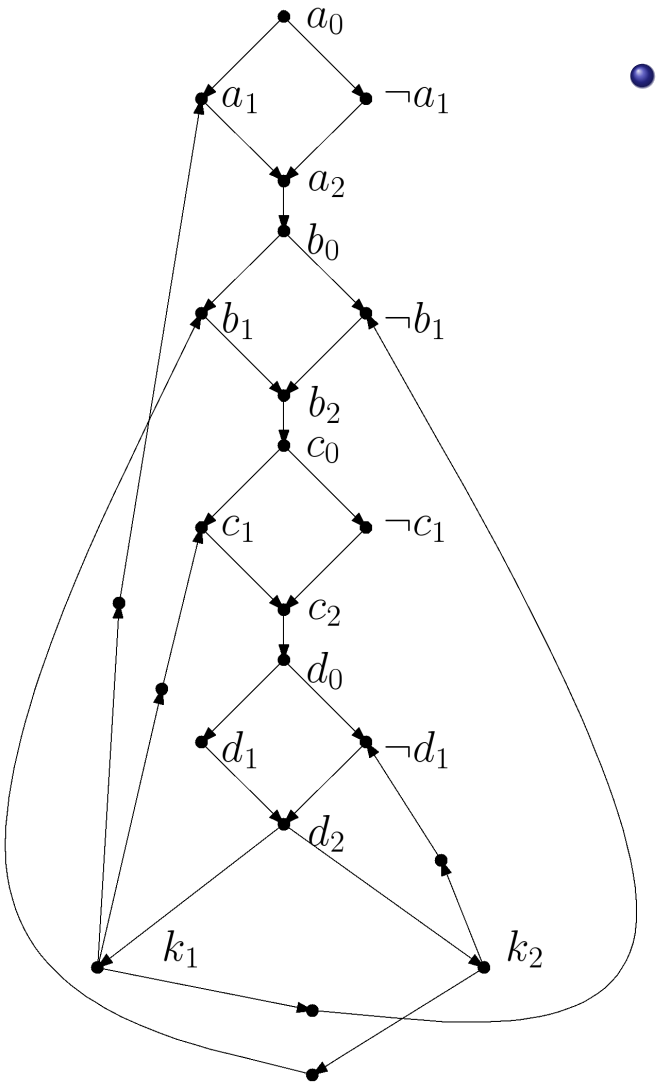
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



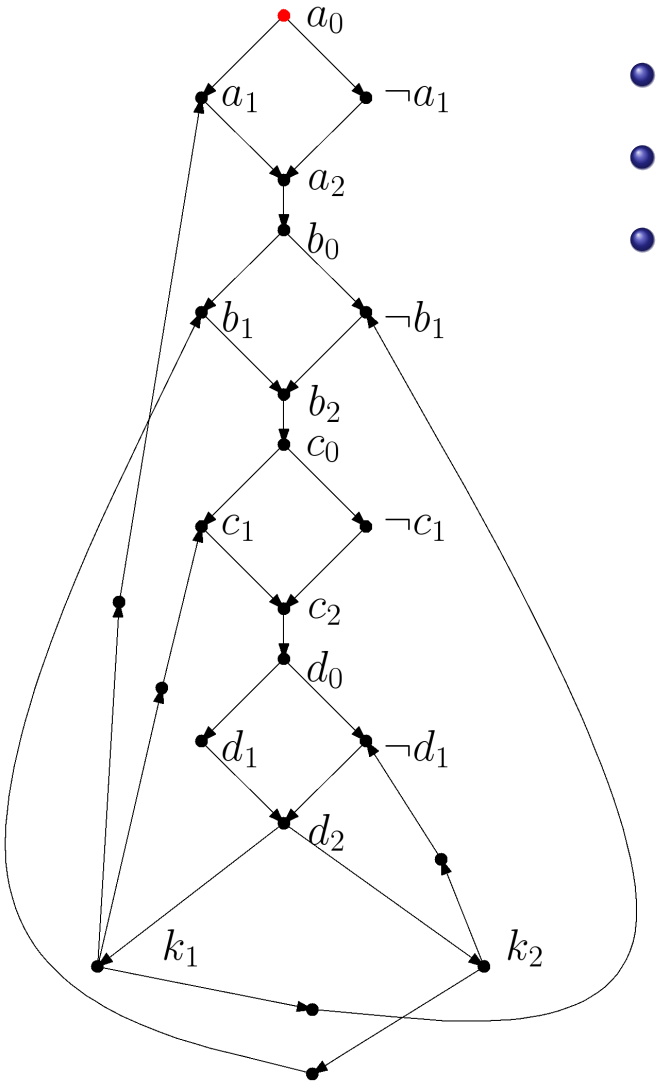
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



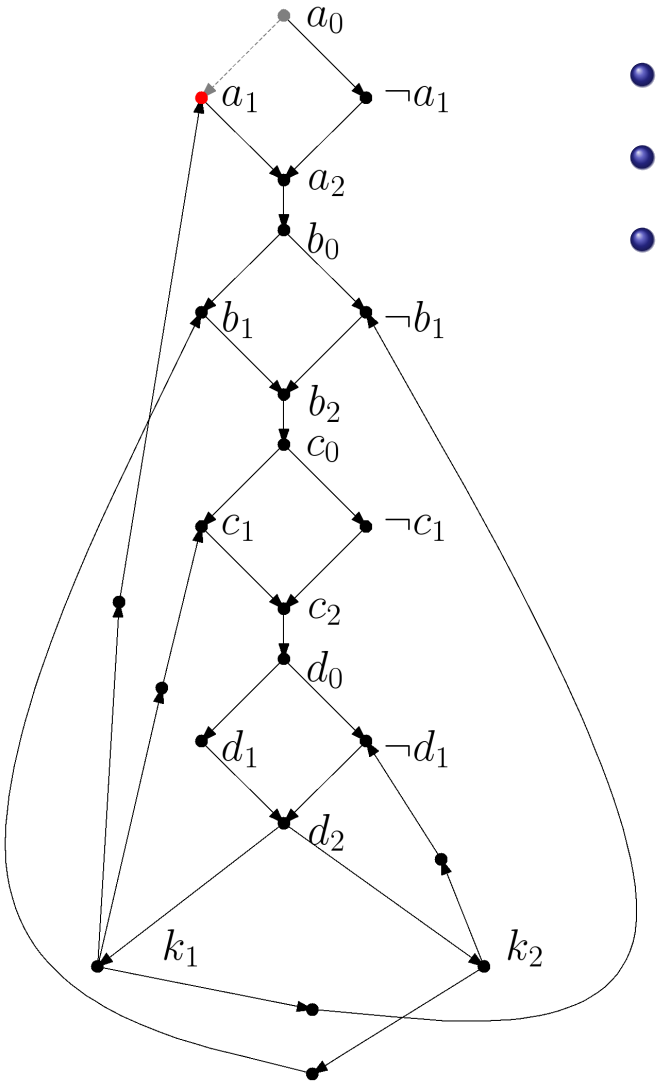
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



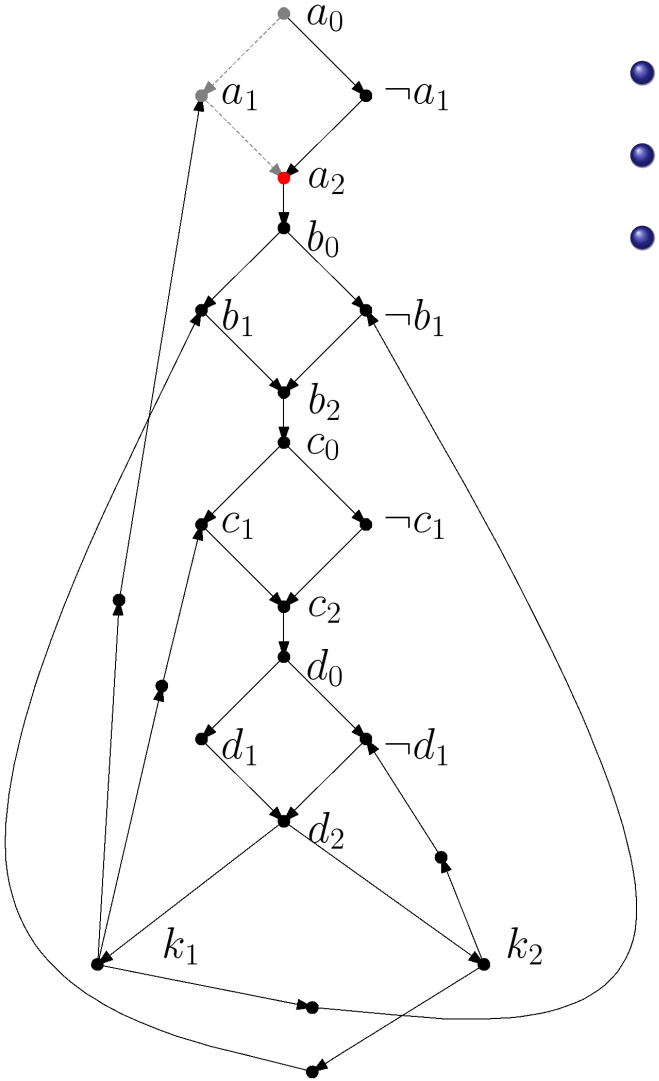
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



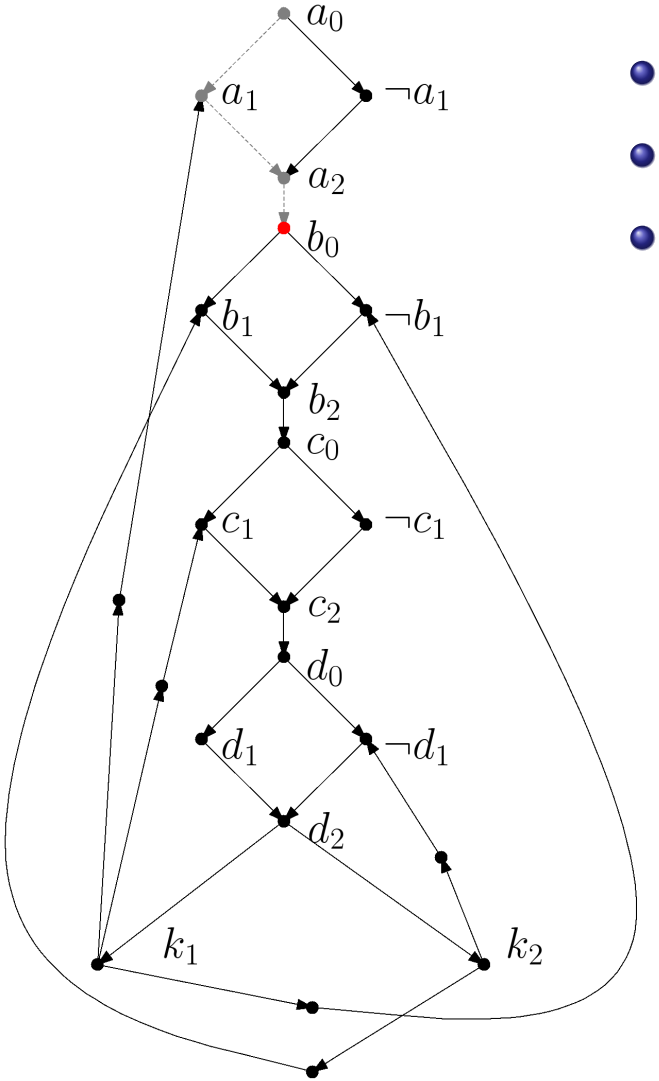
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



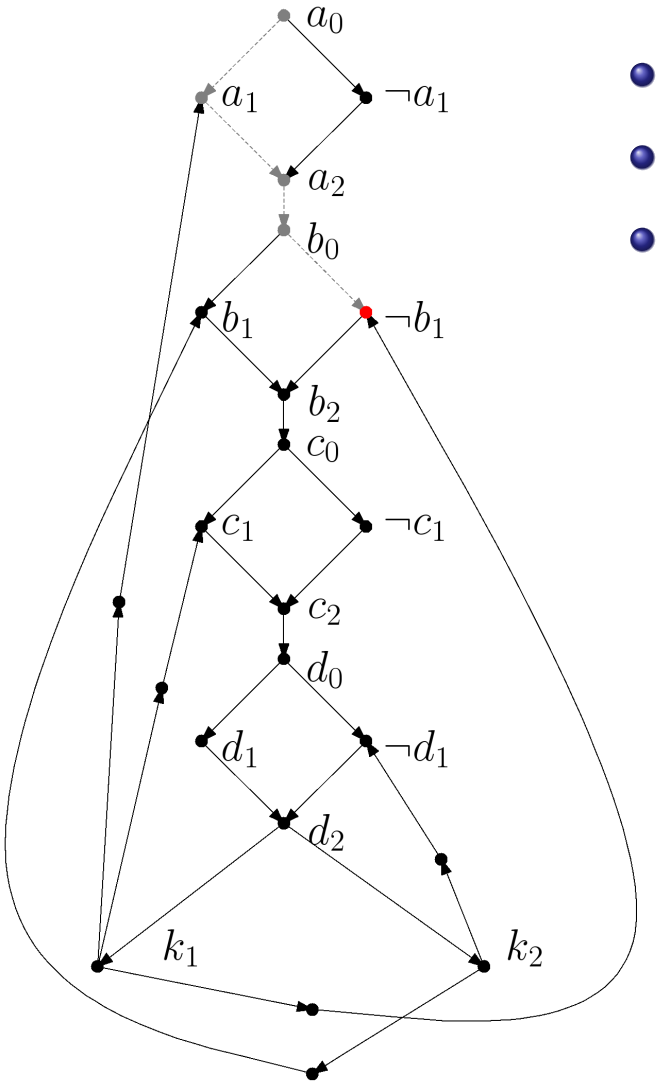
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



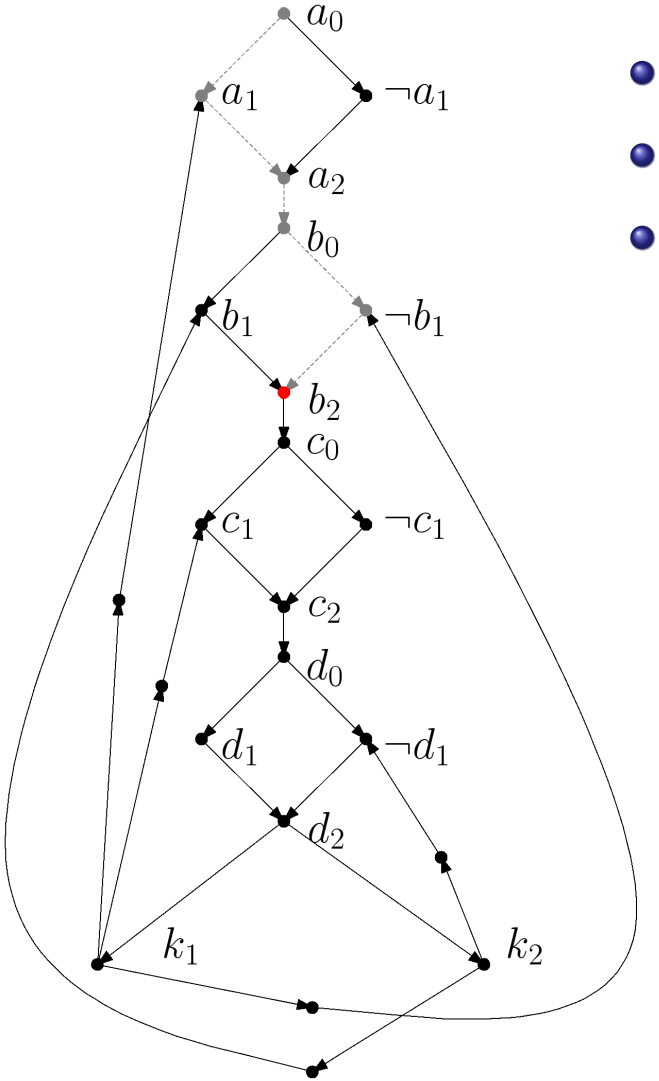
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



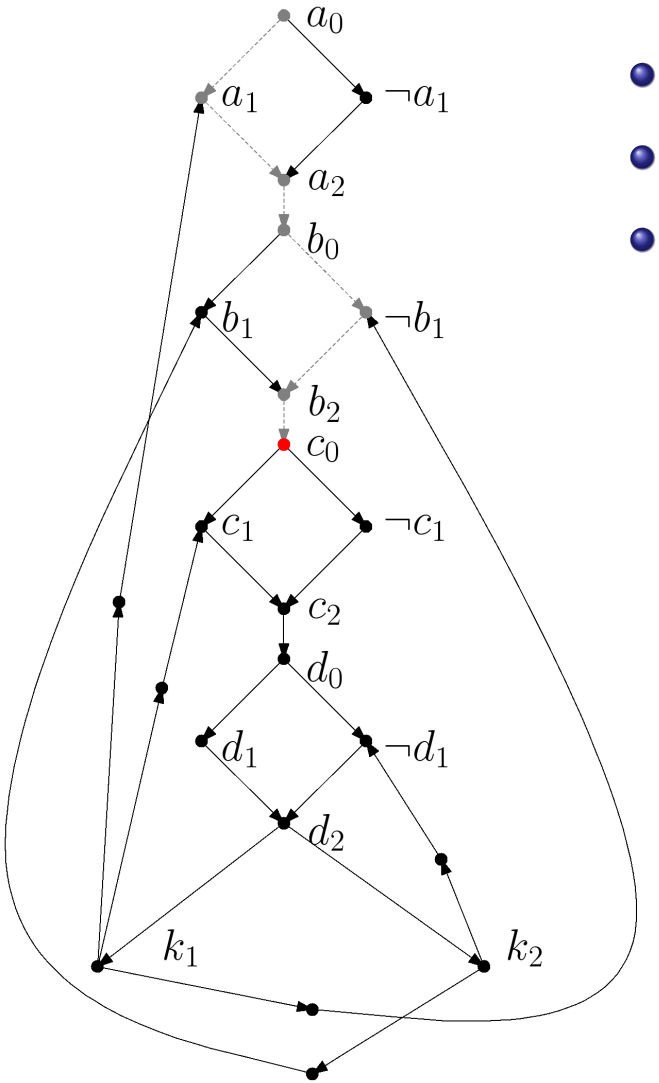
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



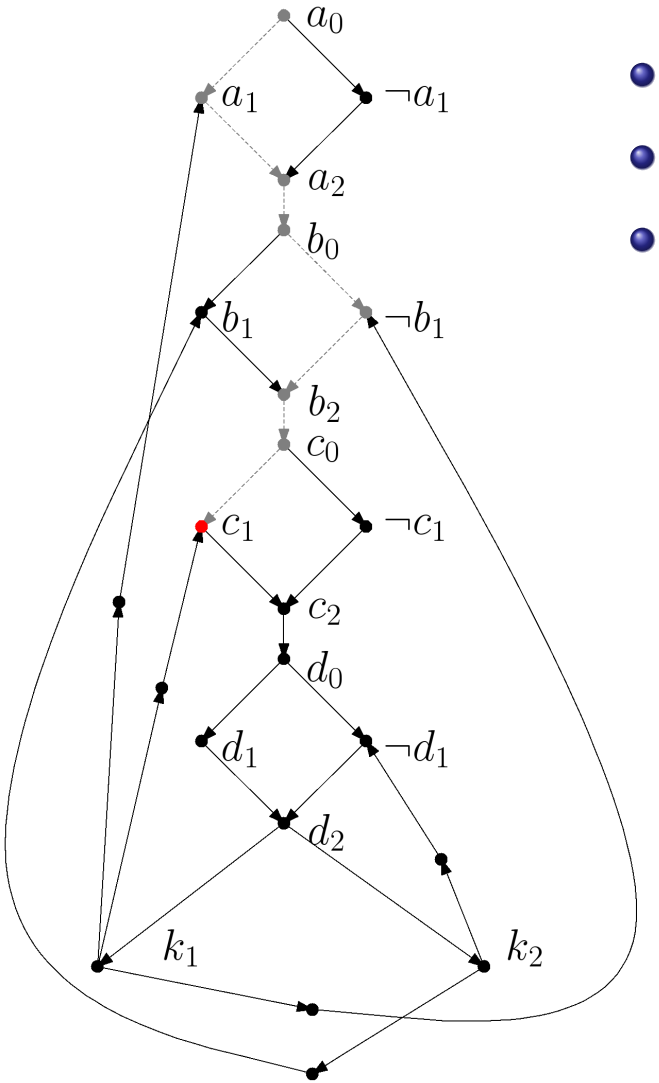
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



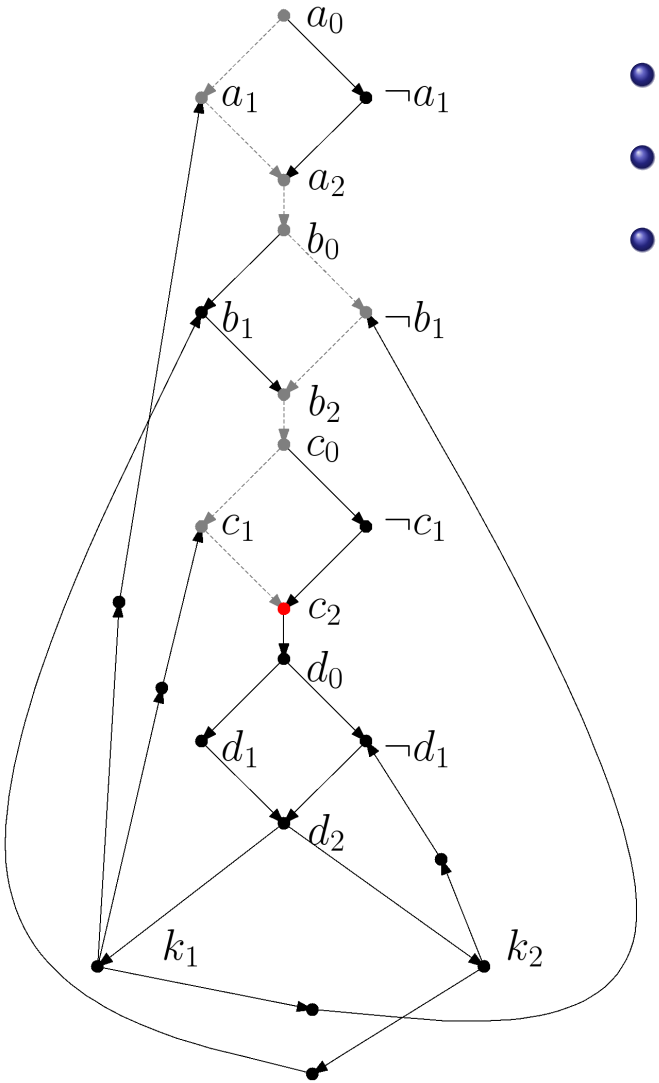
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



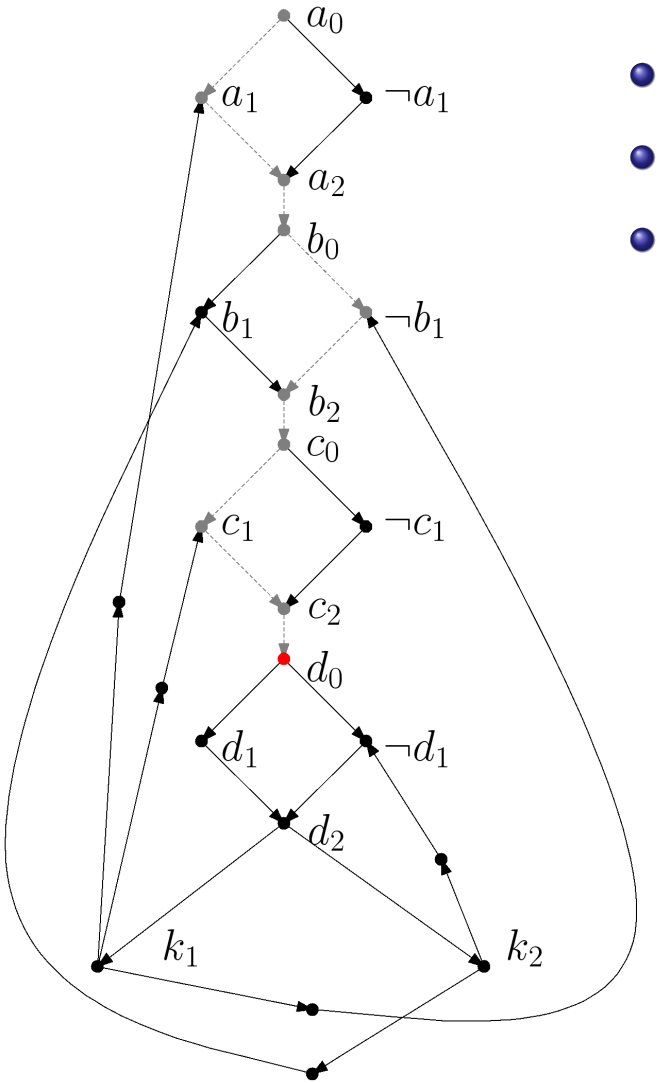
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



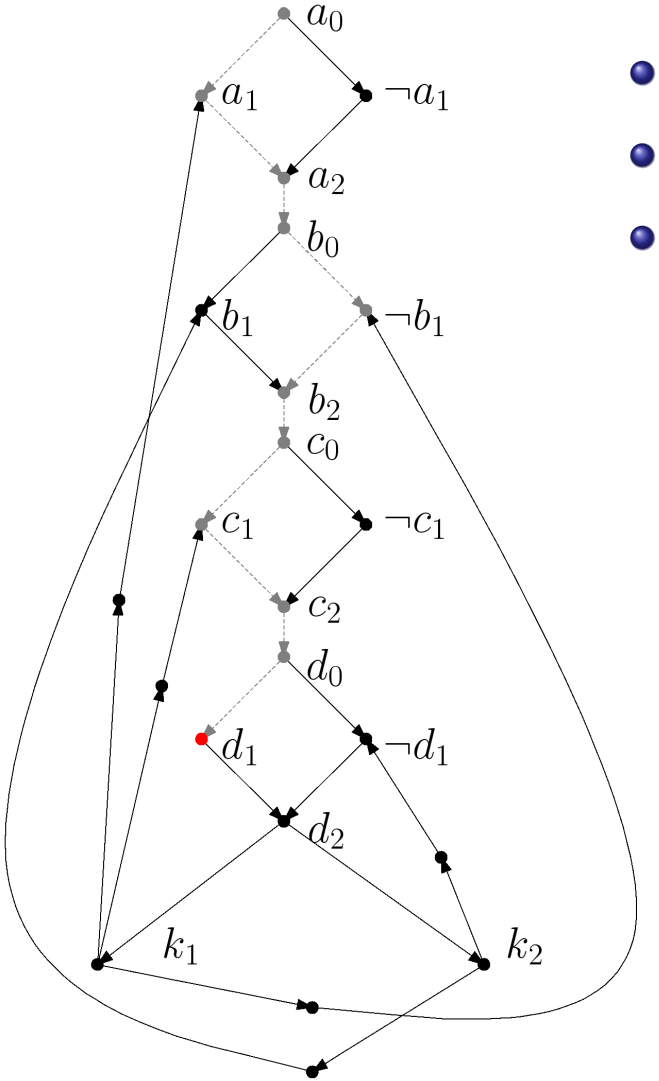
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



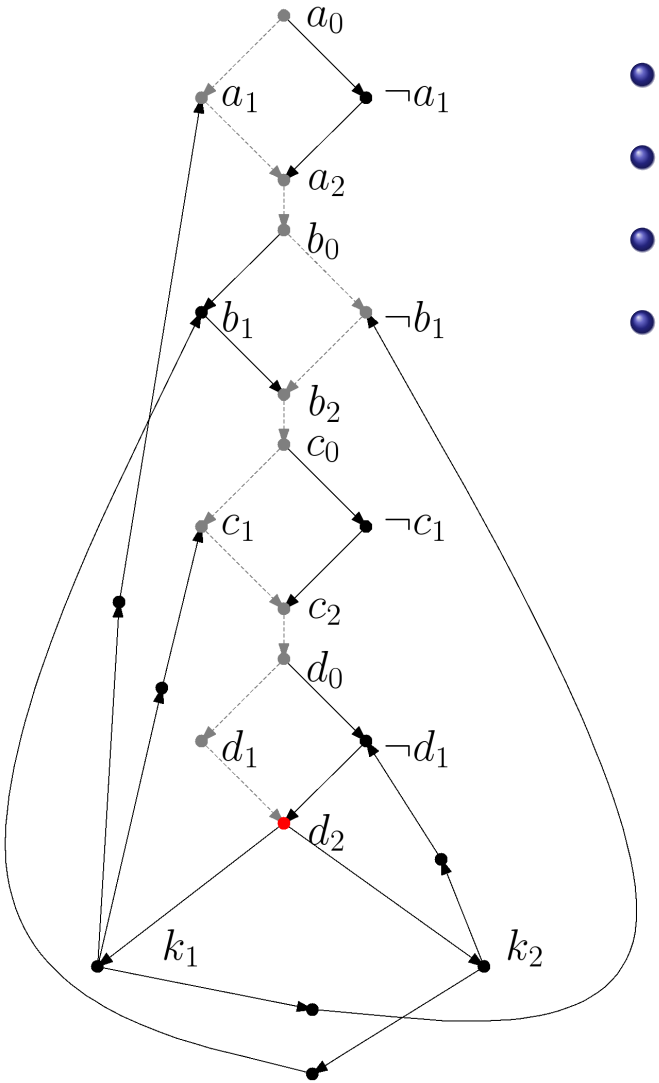
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



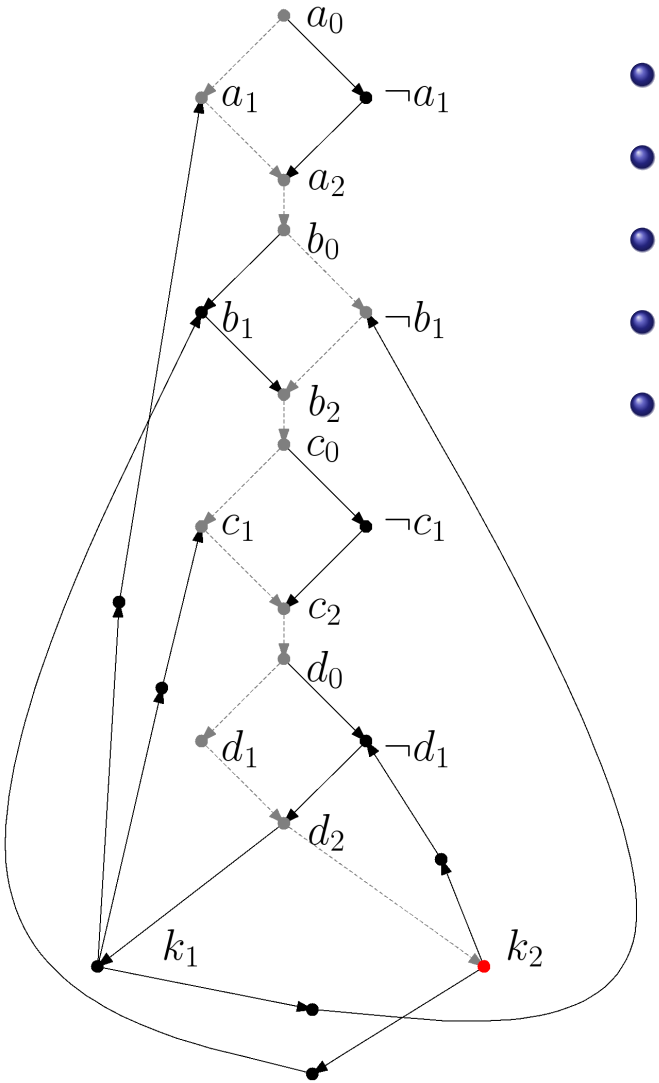
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



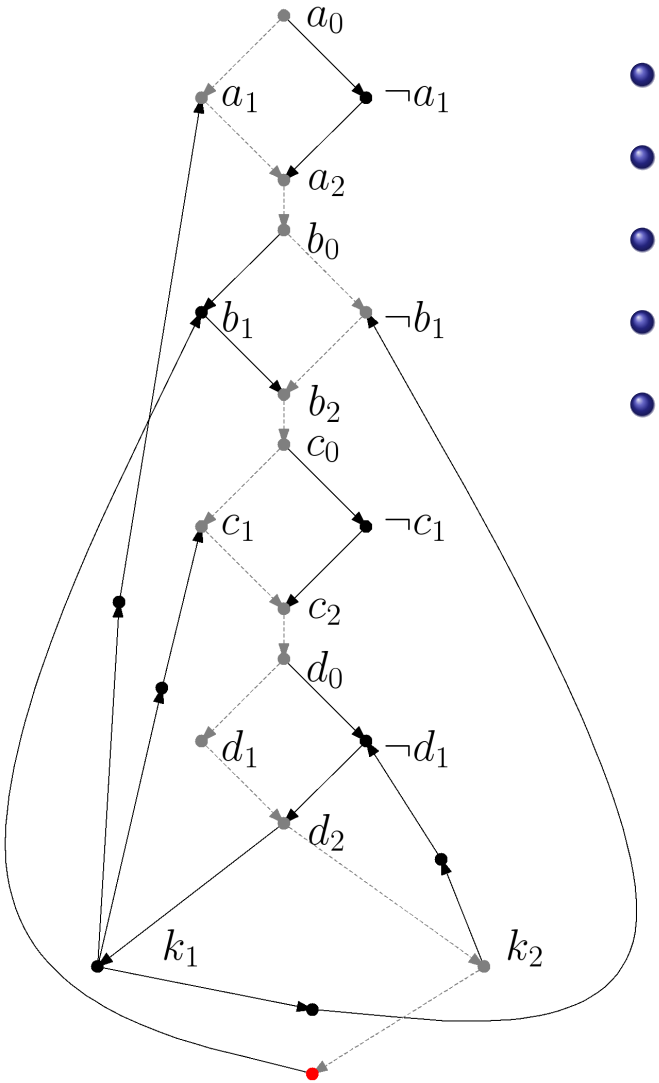
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.
- Spieler 2 wählt eine Klausel.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



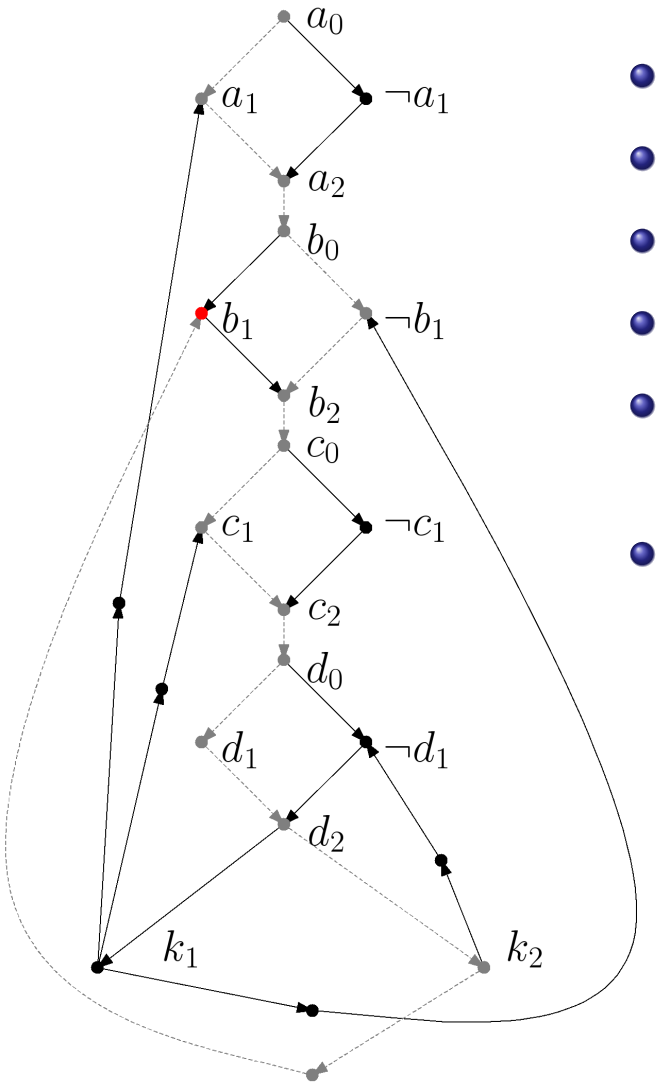
- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.
- Spieler 2 wählt eine Klausel.
- Spieler 1 wählt eine Variable aus dieser Klausel.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.
- Spieler 2 wählt eine Klausel.
- Spieler 1 wählt eine Variable aus dieser Klausel.

Das Problem GG
GG ist PSPACE-schwer (Fortsetzung Beweis)



- $B = \exists a \forall b \exists c \forall d : (a \vee \neg b \vee c) \wedge (b \vee \neg d)$
- Spieler 1 wählt Belegung der $\exists$-Variablen.
- Spieler 2 wählt Belegung der $\forall$-Variablen.
- Spieler 2 wählt eine Klausel.
- Spieler 1 wählt eine Variable aus dieser Klausel.
- Spieler 1 gewinnt genau dann, wenn $B \in TQBF$.

□

Das Problem PGG
Definition, Satz

- GG kann natürlich auch auf planaren Graphen gespielt werden.
- Ein Graph ist planar, wenn er in der Ebene gezeichnet werden kann, ohne dass sich die Kanten überschneiden.

- GG kann natürlich auch auf planaren Graphen gespielt werden.
- Ein Graph ist planar, wenn er in der Ebene gezeichnet werden kann, ohne dass sich die Kanten überschneiden.

Satz

Das Problem, den Gewinner eines GG-Spiels zu bestimmen, ist PSPACE-schwer, auch wenn der Graph planar und bipartit ist und jede Ecke höchstens Grad 3 hat.

- GG kann natürlich auch auf planaren Graphen gespielt werden.
- Ein Graph ist planar, wenn er in der Ebene gezeichnet werden kann, ohne dass sich die Kanten überschneiden.

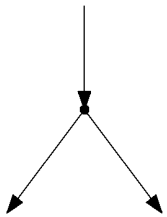
Satz

Das Problem, den Gewinner eines GG-Spiels zu bestimmen, ist PSPACE-schwer, auch wenn der Graph planar und bipartit ist und jede Ecke höchstens Grad 3 hat.

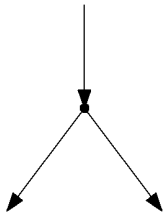
- Ein wie beschrieben konstruierter GG-Graph kann systematisch in einen planaren, bipartiten Graphen überführt werden, in dem jede Ecke höchstens Grad 3 hat.

In einem solchen PGG-Graphen kann man folgende Typen von Ecken unterscheiden:

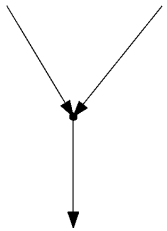
- Wahlmöglichkeit für Spieler 1



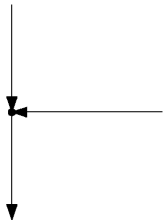
- Wahlmöglichkeit für Spieler 2



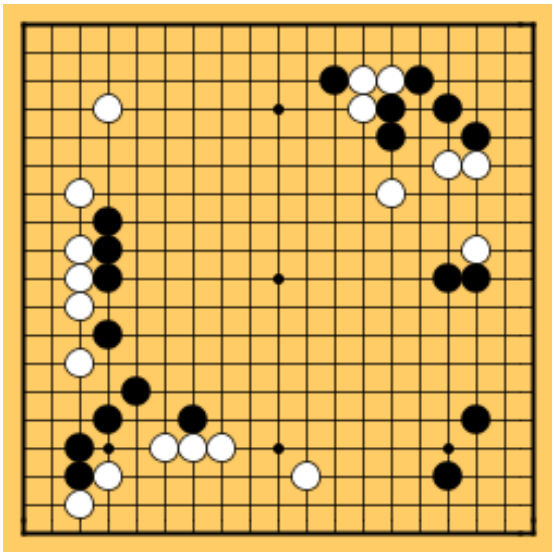
- Zusammenführung



- Test

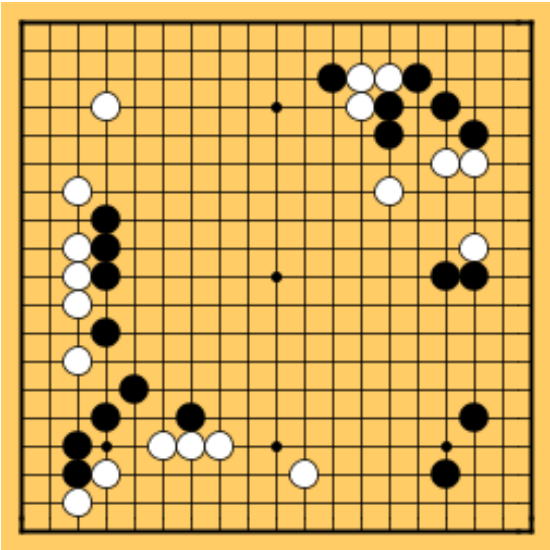


(Triviale Ecken lassen wir aus.)



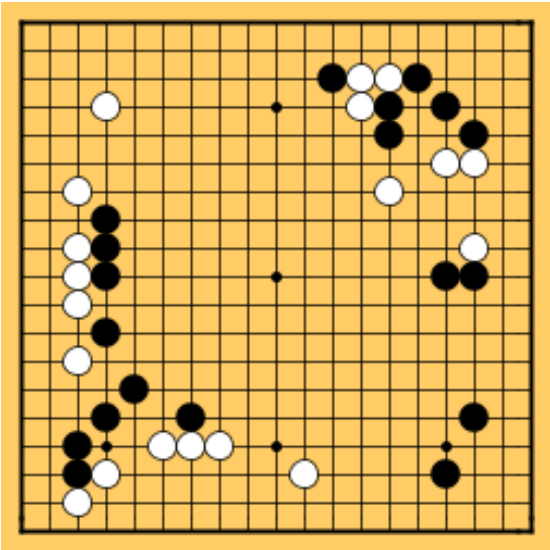
- GO wird auf einem Brett gespielt, auf dem ein Gitter mit $n \times n$ Linien eingezeichnet ist.
- Die Schnittpunkte des Gitters heißen *Felder*.

Die GO-Regeln
Grundlegendes



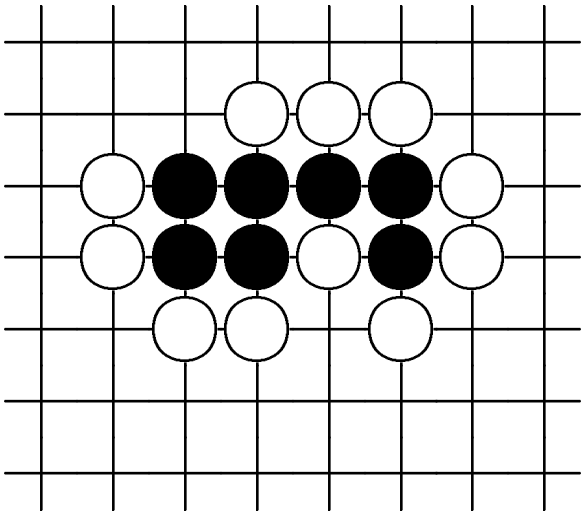
- GO wird auf einem Brett gespielt, auf dem ein Gitter mit $n \times n$ Linien eingezeichnet ist.
- Die Schnittpunkte des Gitters heißen *Felder*.
- Es gibt zwei Spieler, Weiß und Schwarz.
- Die Spieler ziehen abwechselnd, indem sie einen ihrer Steine auf ein freies Feld setzen oder passen.

Die GO-Regeln
Grundlegendes



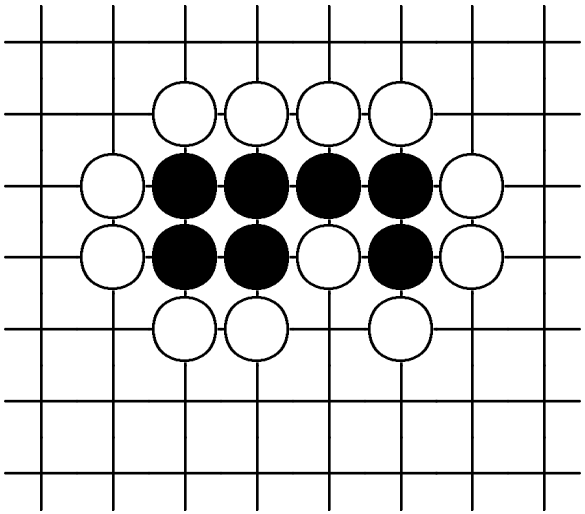
- GO wird auf einem Brett gespielt, auf dem ein Gitter mit $n \times n$ Linien eingezeichnet ist.
- Die Schnittpunkte des Gitters heißen *Felder*.
- Es gibt zwei Spieler, Weiß und Schwarz.
- Die Spieler ziehen abwechselnd, indem sie einen ihrer Steine auf ein freies Feld setzen oder passen.
- Eine maximale Menge von gleichfarbigen, benachbarten Steinen heißt *Gruppe*.

Die GO-Regeln
Schlagen von Steinen



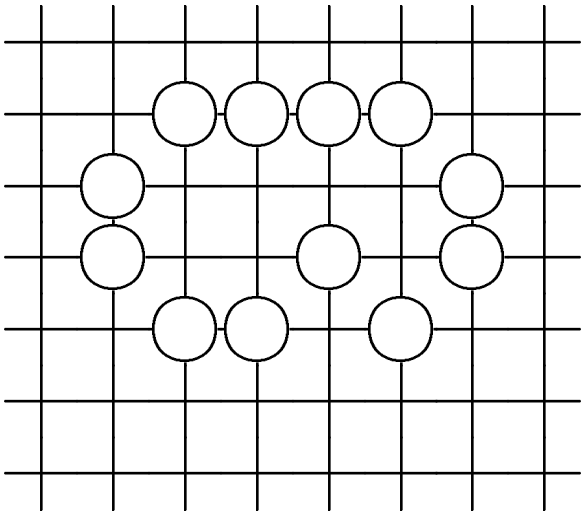
- Wenn eine Gruppe von schwarzen Steinen vollständig von weißen Steinen umgeben ist, wird sie vom Brett entfernt.
- (Und andersrum)

Die GO-Regeln
Schlagen von Steinen



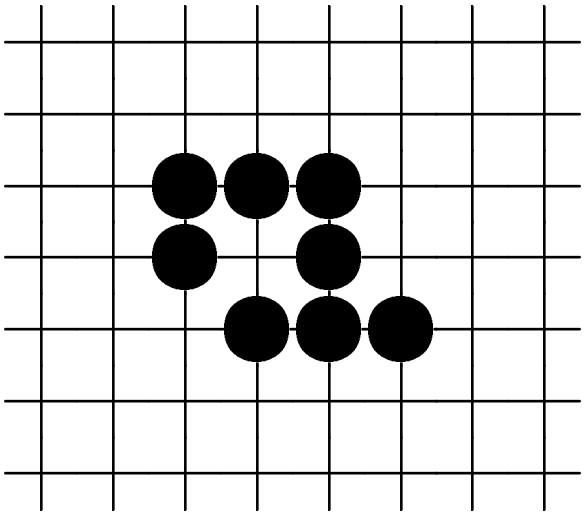
- Wenn eine Gruppe von schwarzen Steinen vollständig von weißen Steinen umgeben ist, wird sie vom Brett entfernt.
- (Und andersrum)

Die GO-Regeln
Schlagen von Steinen



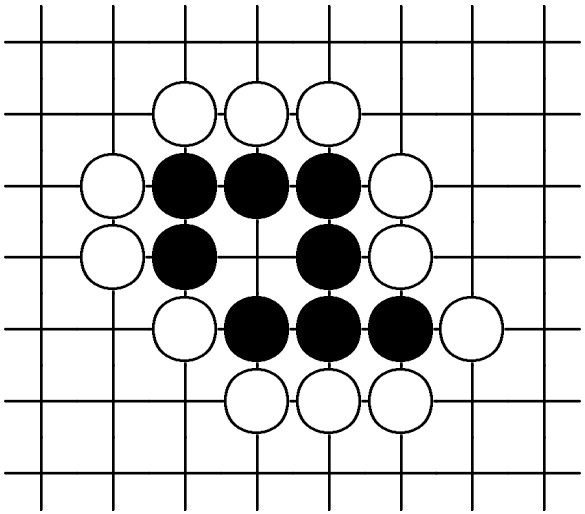
- Wenn eine Gruppe von schwarzen Steinen vollständig von weißen Steinen umgeben ist, wird sie vom Brett entfernt.
- (Und andersrum)

Die GO-Regeln
Selbstmord



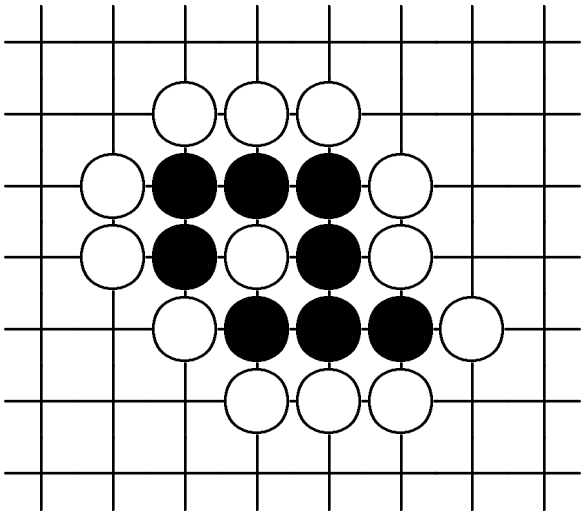
- Es ist nicht erlaubt, einen Stein so zu setzen, dass er (bzw. die zugehörige Gruppe) nach dem Spielzug kein freies benachbartes Feld mehr hat.

Die GO-Regeln
Selbstmord



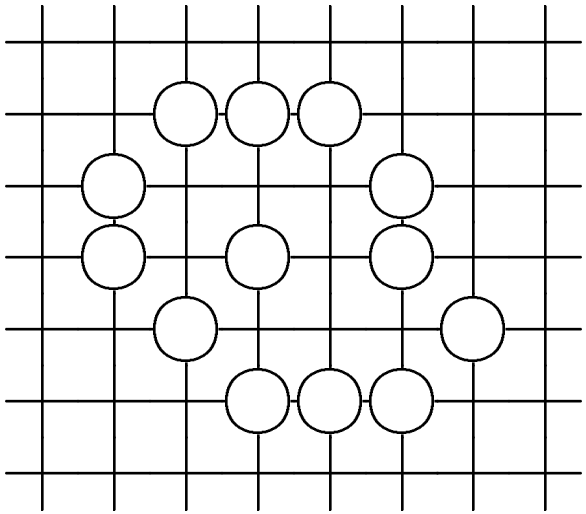
- Es ist nicht erlaubt, einen Stein so zu setzen, dass er (bzw. die zugehörige Gruppe) nach dem Spielzug kein freies benachbartes Feld mehr hat.
- Ausnahme: Werden durch das Setzen die umliegenden Steine geschlagen, ist der Zug erlaubt.

Die GO-Regeln
Selbstmord

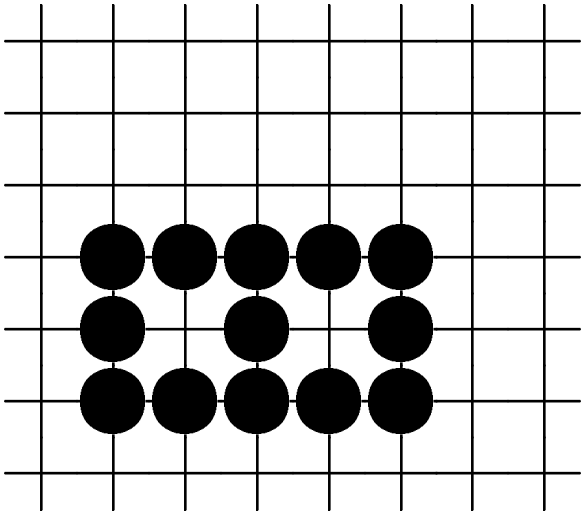


- Es ist nicht erlaubt, einen Stein so zu setzen, dass er (bzw. die zugehörige Gruppe) nach dem Spielzug kein freies benachbartes Feld mehr hat.
- Ausnahme: Werden durch das Setzen die umliegenden Steine geschlagen, ist der Zug erlaubt.

Die GO-Regeln
Selbstmord

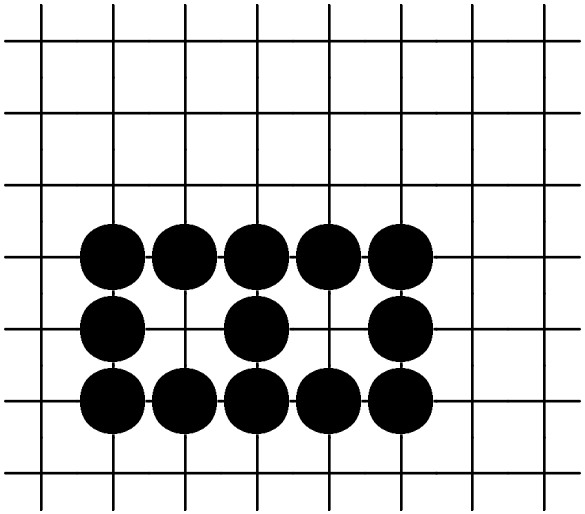


- Es ist nicht erlaubt, einen Stein so zu setzen, dass er (bzw. die zugehörige Gruppe) nach dem Spielzug kein freies benachbartes Feld mehr hat.
- Ausnahme: Werden durch das Setzen die umliegenden Steine geschlagen, ist der Zug erlaubt.



- Bestimmte Gruppen können nicht geschlagen werden.
- Nämlich solche, die in ihrem Inneren mindestens zwei freie, nicht benachbarte Felder (*Augen*) haben.

Die GO-Regeln
Unschlagbare Gruppen



- Bestimmte Gruppen können nicht geschlagen werden.
- Nämlich solche, die in ihrem Inneren mindestens zwei freie, nicht benachbarte Felder (*Augen*) haben.
- Das Schlagen einer solchen Gruppe ist durch die Selbstmord-Regel ausgeschlossen.
- Die freien Felder in einer solchen Gruppe werden *Gebiet* genannt.

- Das Spiel ist beendet, wenn beide Spieler passen.
- Es werden alle Steine entfernt, die nicht zu einer unschlagbaren Gruppe gehören.

- Das Spiel ist beendet, wenn beide Spieler passen.
- Es werden alle Steine entfernt, die nicht zu einer unschlagbaren Gruppe gehören.
- Die Punktzahl für Weiß ist die Anzahl der freien Felder in den unschlagbaren weißen Gruppen, abzüglich der Anzahl im Spielverlauf geschlagenen weißen Steine.
- Die Punktzahl für Schwarz wird analog berechnet.
- Der Spieler mit der höheren Punktzahl gewinnt.

Satz (Lichtenstein & Sipser, 1980)

Das Problem, für eine beliebige GO-Situation den Gewinner zu bestimmen, ist PSPACE-schwer.

Satz (Lichtenstein & Sipser, 1980)

Das Problem, für eine beliebige GO-Situation den Gewinner zu bestimmen, ist PSPACE-schwer.

Beweis:

- Idee: Reduktion PGG $\leq$ GO.
- Gegeben ein Graph, der wie bei der Reduktion TQBF $\leq$ GG konstruiert wurde und dann in einen planaren, bipartiten Graphen mit Eckengrad höchstens 3 überführt wurde.

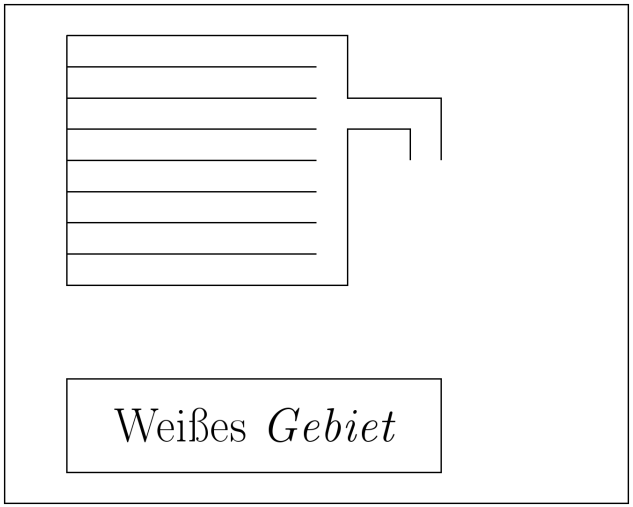
Satz (Lichtenstein & Sipser, 1980)

Das Problem, für eine beliebige GO-Situation den Gewinner zu bestimmen, ist PSPACE-schwer.

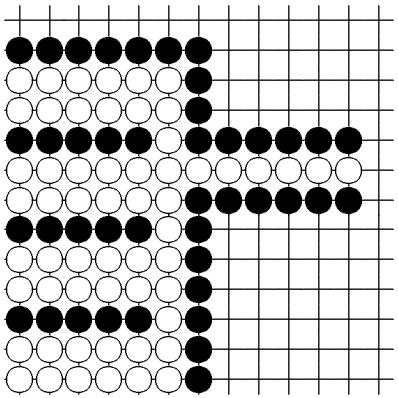
Beweis:

- Idee: Reduktion PGG $\leq$ GO.
- Gegeben ein Graph, der wie bei der Reduktion TQBF $\leq$ GG konstruiert wurde und dann in einen planaren, bipartiten Graphen mit Eckengrad höchstens 3 überführt wurde.
- Zu diesem Graphen konstruieren wir ein GO-Spiel, in dem Schwarz genau dann gewinnt, wenn Spieler 1 das PGG-Spiel gewinnt.

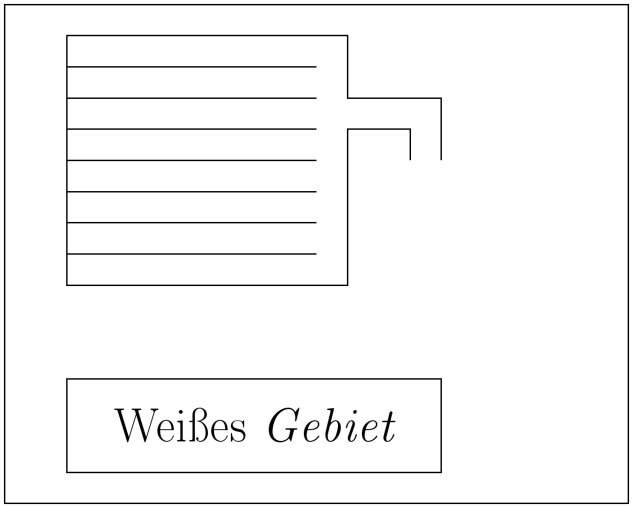
Konstruktion der GO-Situation
Globale Struktur

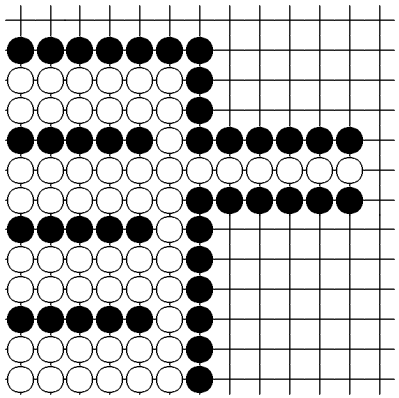


- Auf dem GO-Brett gebe es eine unschlagbare weiße Gruppe, mit sehr großem *Gebiet*.



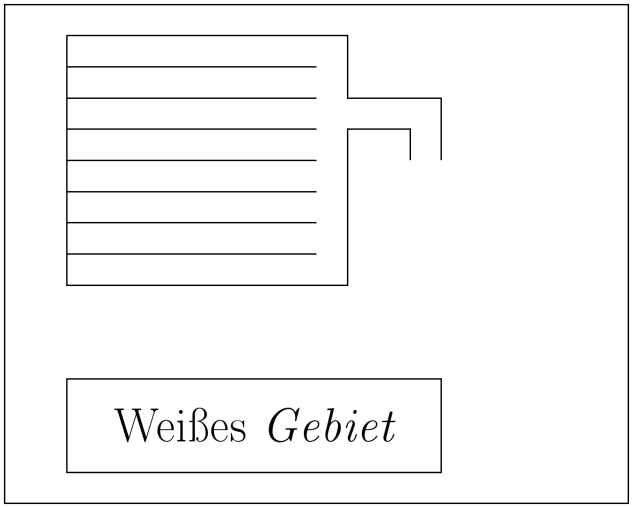
Konstruktion der GO-Situation
Globale Struktur

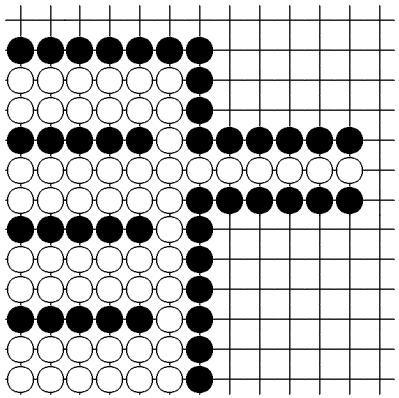




- Auf dem GO-Brett gebe es eine unschlagbare weiße Gruppe, mit sehr großem *Gebiet*.
- Außerdem gebe es eine noch größere weiße Gruppe, die von Schwarz fast vollständig umgeben ist. Diese Gruppe sei so groß, dass Schwarz gewinnt, wenn er sie umgeben kann.

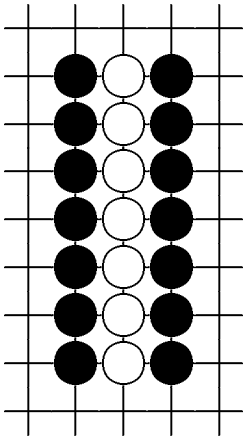
Konstruktion der GO-Situation
Globale Struktur

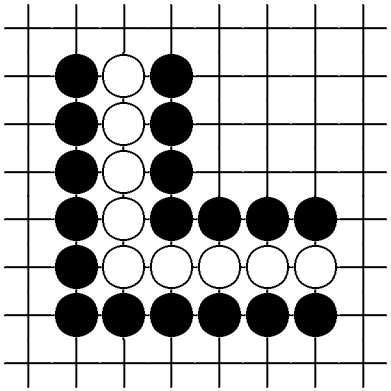




- Auf dem GO-Brett gebe es eine unschlagbare weiße Gruppe, mit sehr großem *Gebiet*.
- Außerdem gebe es eine noch größere weiße Gruppe, die von Schwarz fast vollständig umgeben ist. Diese Gruppe sei so groß, dass Schwarz gewinnt, wenn er sie umgeben kann.
- Weiß versucht, die schlagbare Gruppe mit einer unschlagbaren Gruppe zu verbinden.
- Schwarz versucht, die schlagbare Gruppe vollständig zu umgeben.

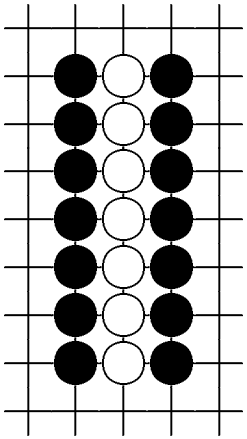
Konstruktion der GO-Situation
Lokale Struktur

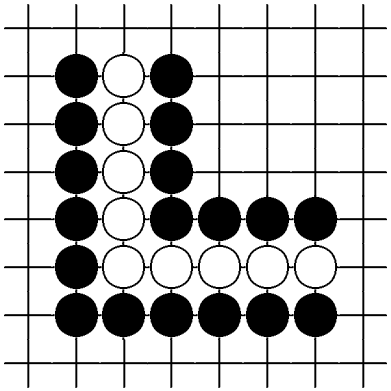




- Zu jeder Kante und jeder Ecke des PGG-Graphen wird eine bestimmte Konfiguration von Steinen auf dem Brett eingefügt.

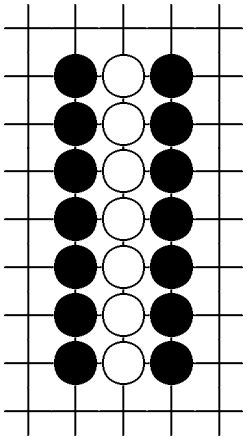
Konstruktion der GO-Situation
Lokale Struktur

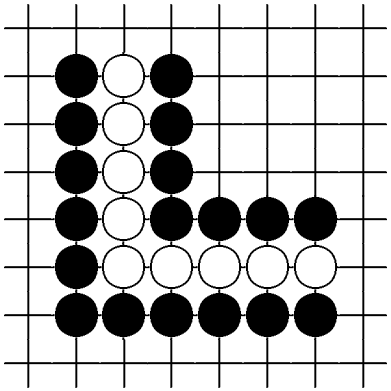




- Zu jeder Kante und jeder Ecke des PGG-Graphen wird eine bestimmte Konfiguration von Steinen auf dem Brett eingefügt.
- Jede Kante im PGG-Graphen wird durch eine Röhre dargestellt.
- Jede Ecke im PGG-Graphen wird durch eine Abzweigung dargestellt.

Konstruktion der GO-Situation
Lokale Struktur



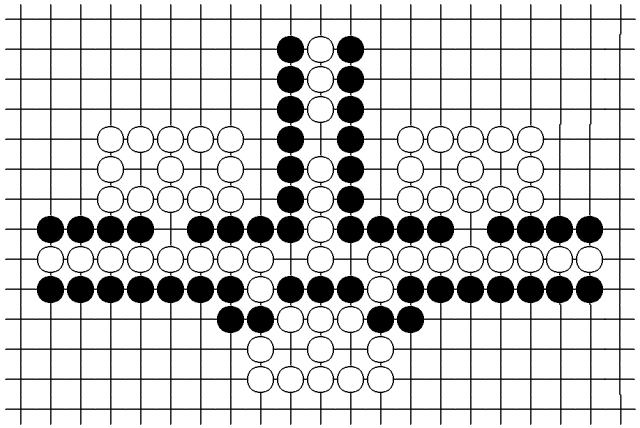


- Zu jeder Kante und jeder Ecke des PGG-Graphen wird eine bestimmte Konfiguration von Steinen auf dem Brett eingefügt.
- Jede Kante im PGG-Graphen wird durch eine Röhre dargestellt.
- Jede Ecke im PGG-Graphen wird durch eine Abzweigung dargestellt.
- Die Abzweigungen unterscheiden sich, je nachdem durch welchen Typ von Ecke sie erzeugt werden.
- Die zur Startecke gehörige Abzweigung schließt an die schlagbare weiße Gruppe an.

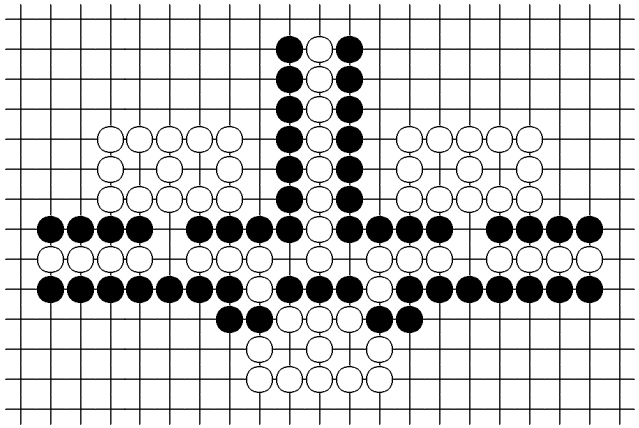
Konstruktion der GO-Situation

Lokale Struktur, Typen von Ecken

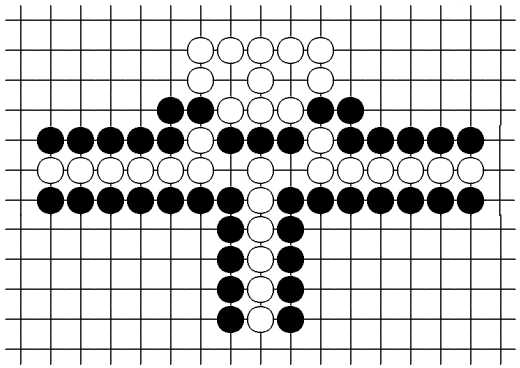
Wahlmöglichkeit für Spieler 1



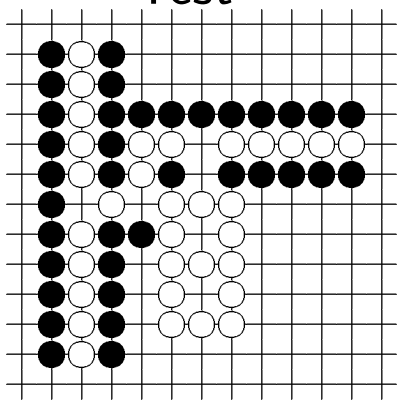
Wahlmöglichkeit für Spieler 2



Zusammenführung



Test



Behauptung

Die konstruierte GO-Situation entspricht dem Spiel auf einem PGG-Graphen, wobei Schwarz genau dann gewinnt, wenn Spieler 1 im PGG-Spiel gewinnt.

Behauptung

Die konstruierte GO-Situation entspricht dem Spiel auf einem PGG-Graphen, wobei Schwarz genau dann gewinnt, wenn Spieler 1 im PGG-Spiel gewinnt.

- Dazu betrachten wir die einzelnen Konfigurationen.

Behauptung

Die konstruierte GO-Situation entspricht dem Spiel auf einem PGG-Graphen, wobei Schwarz genau dann gewinnt, wenn Spieler 1 im PGG-Spiel gewinnt.

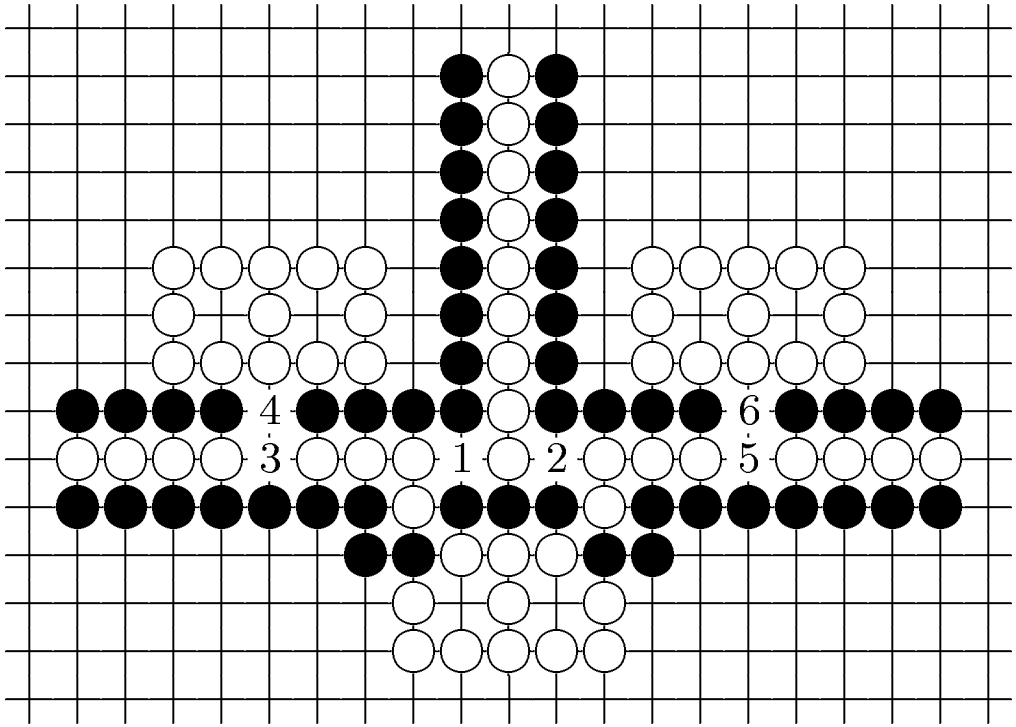
- Dazu betrachten wir die einzelnen Konfigurationen.
- Beim Betreten jeder Konfiguration ist Weiß am Zug.

Behauptung

Die konstruierte GO-Situation entspricht dem Spiel auf einem PGG-Graphen, wobei Schwarz genau dann gewinnt, wenn Spieler 1 im PGG-Spiel gewinnt.

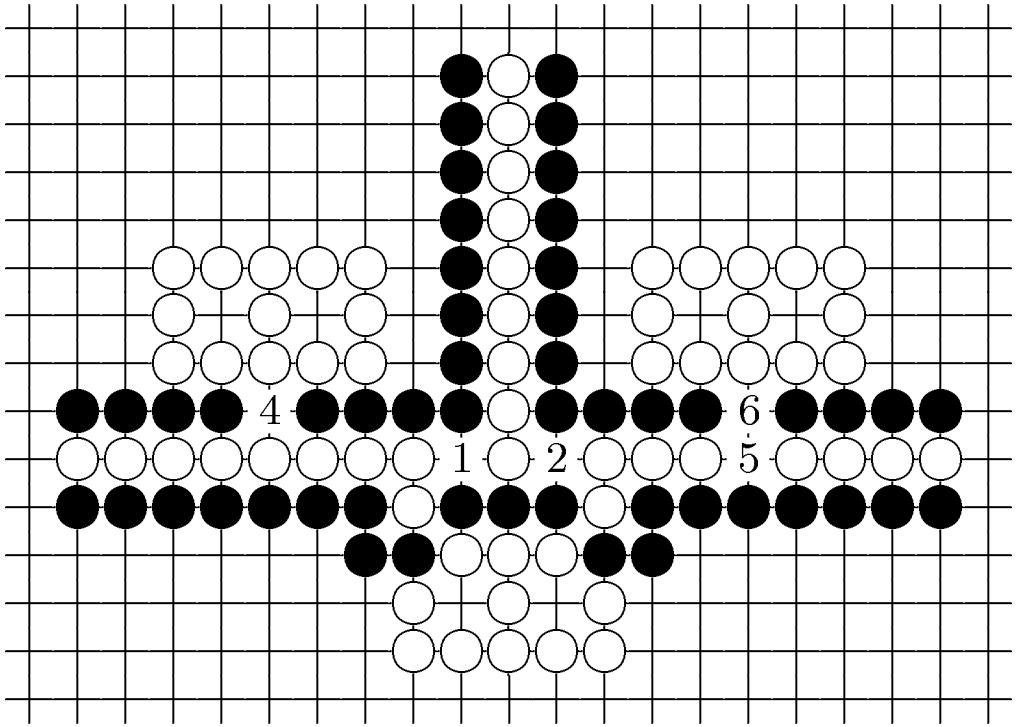
- Dazu betrachten wir die einzelnen Konfigurationen.
- Beim Betreten jeder Konfiguration ist Weiß am Zug.
- Das Brett wird so durchlaufen, wie es auch der PGG-Graph würde.

Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



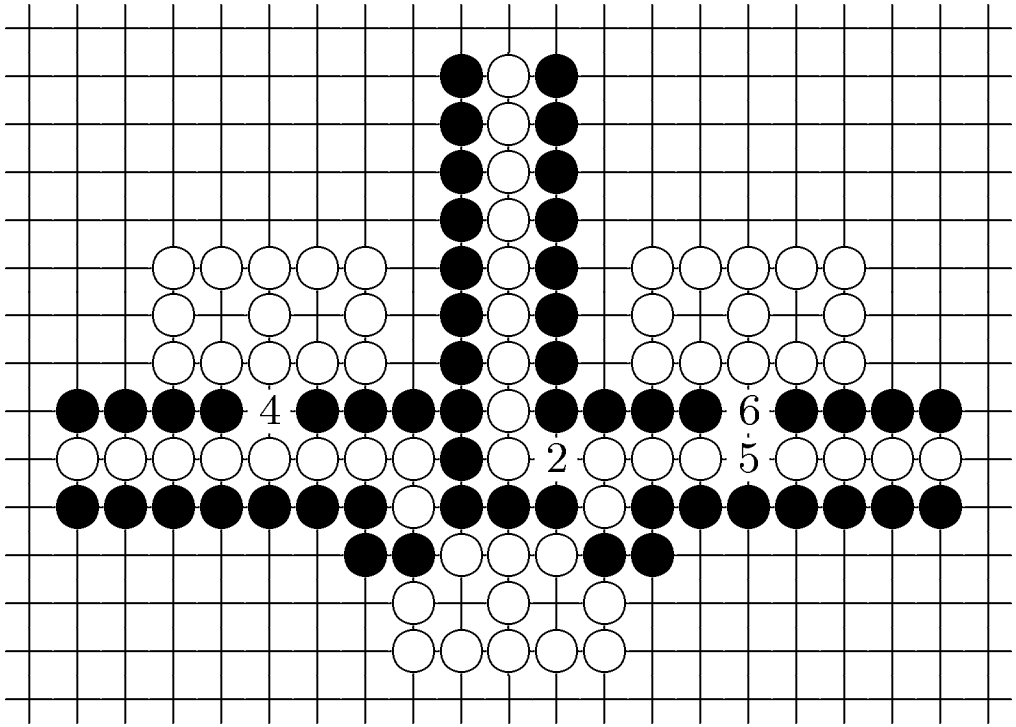
- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.

Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



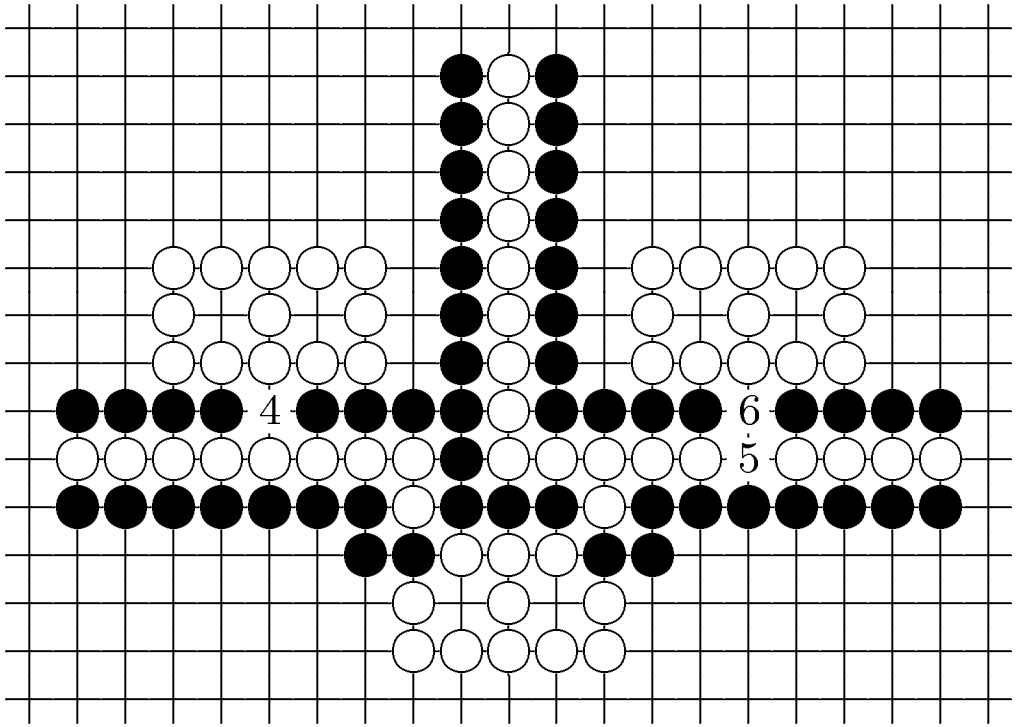
- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.

Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.

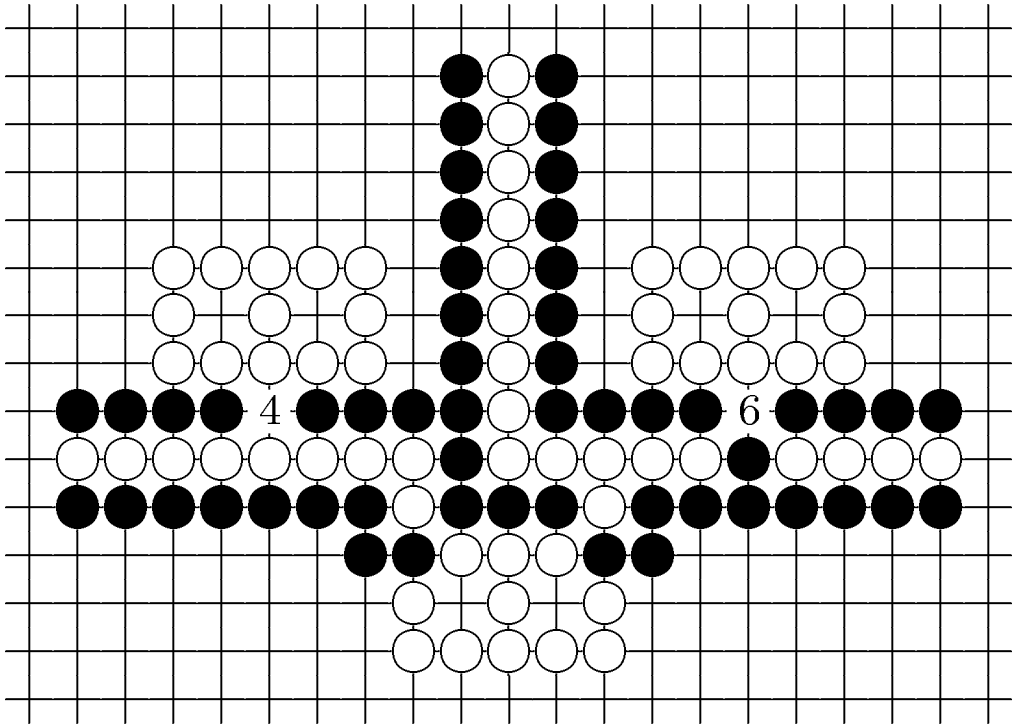
Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.

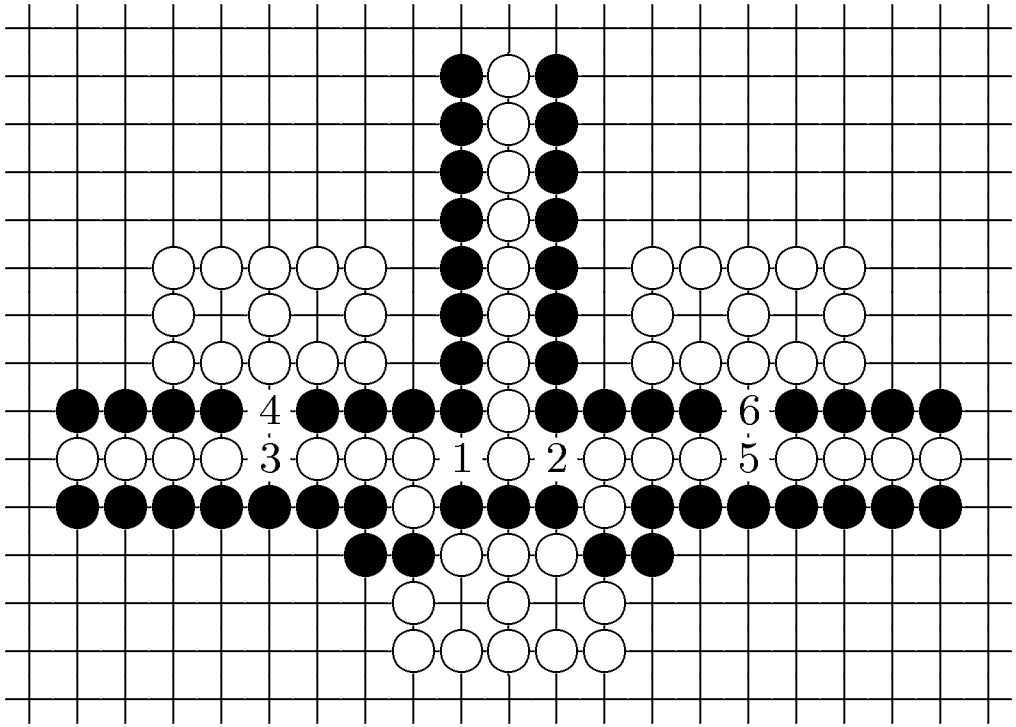
Konfigurationen in der GO-Situation

Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.

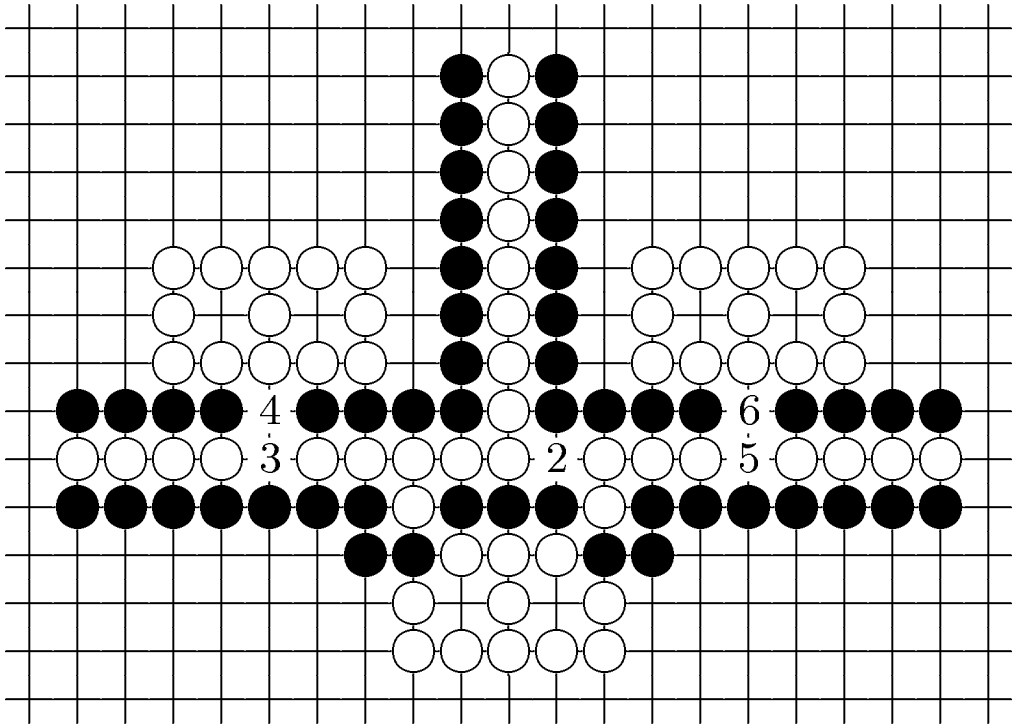
Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.

Konfigurationen in der GO-Situation

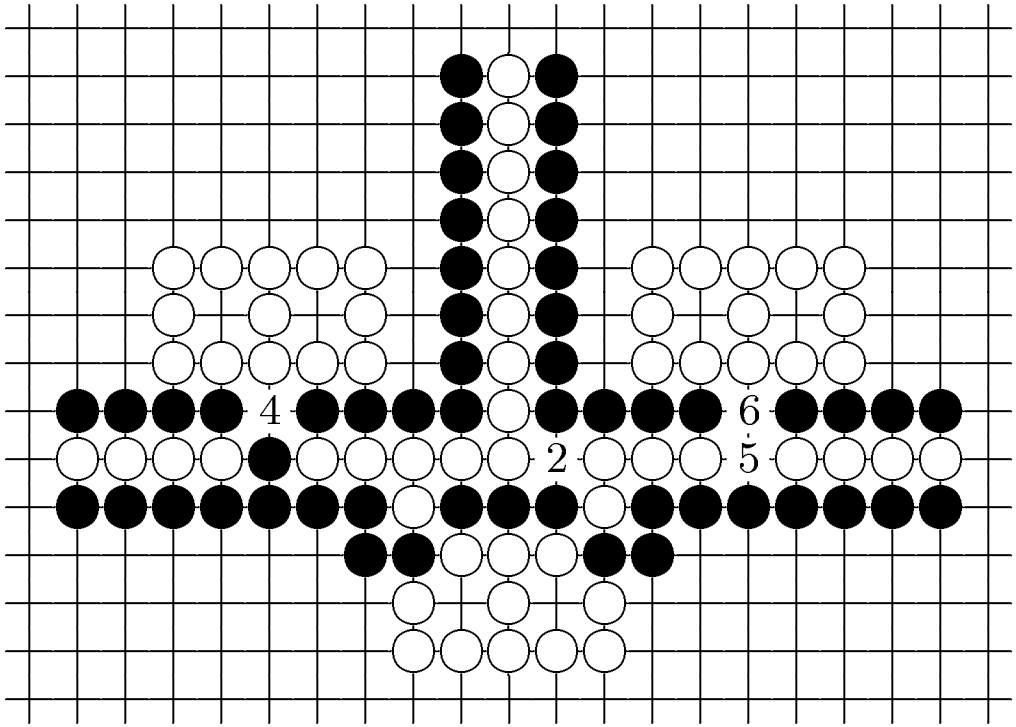
Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.

Konfigurationen in der GO-Situation

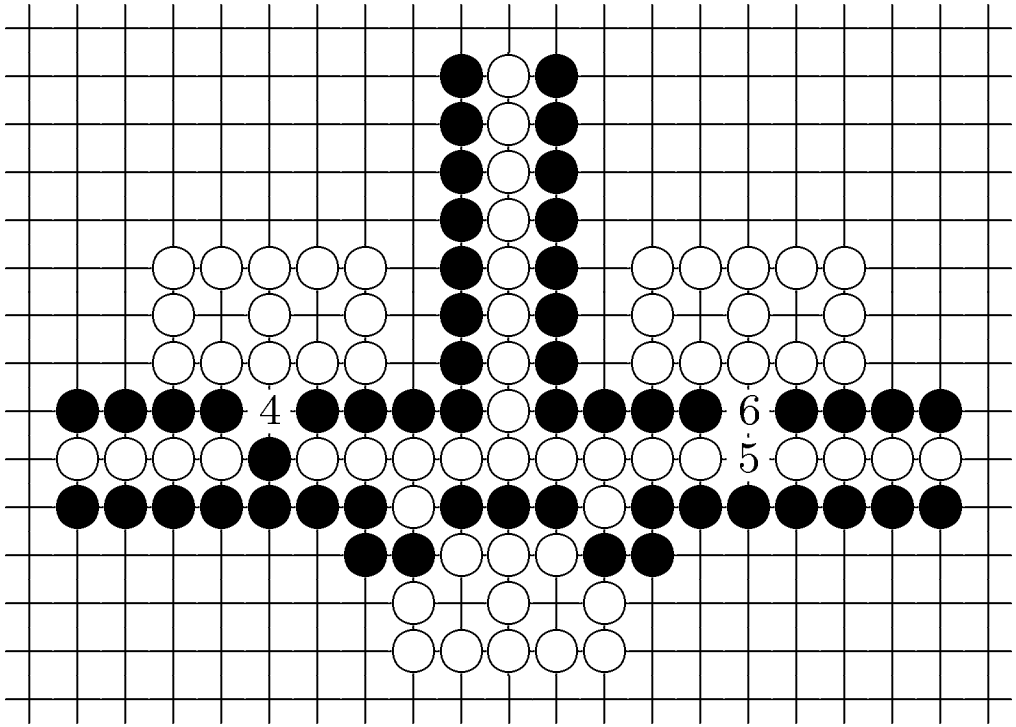
Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.

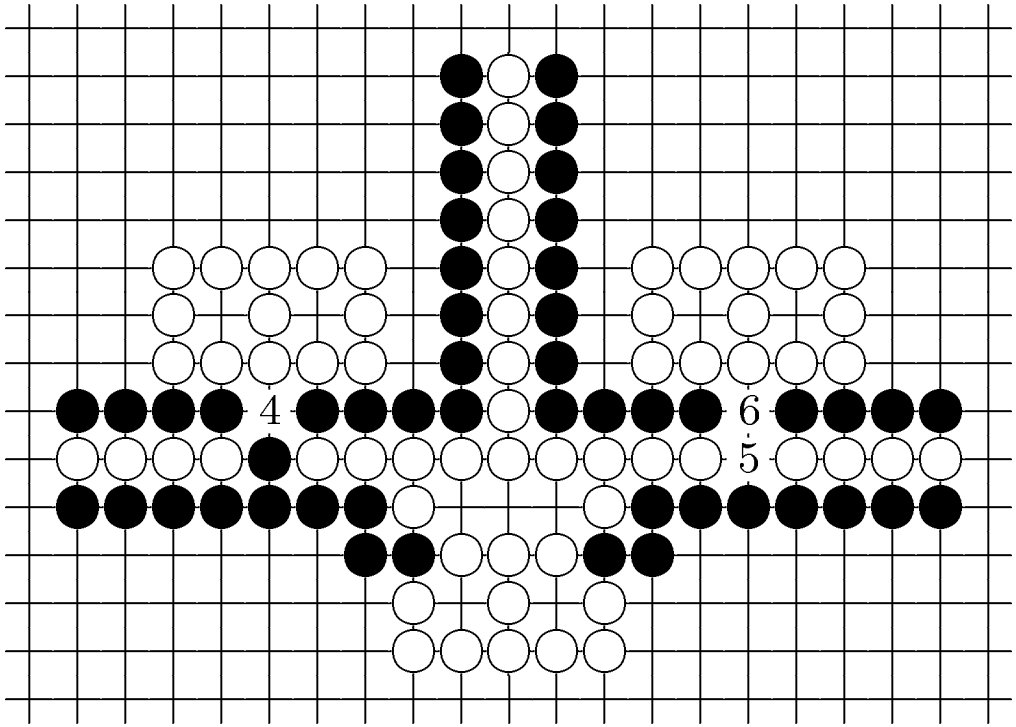
Konfigurationen in der GO-Situation

Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.

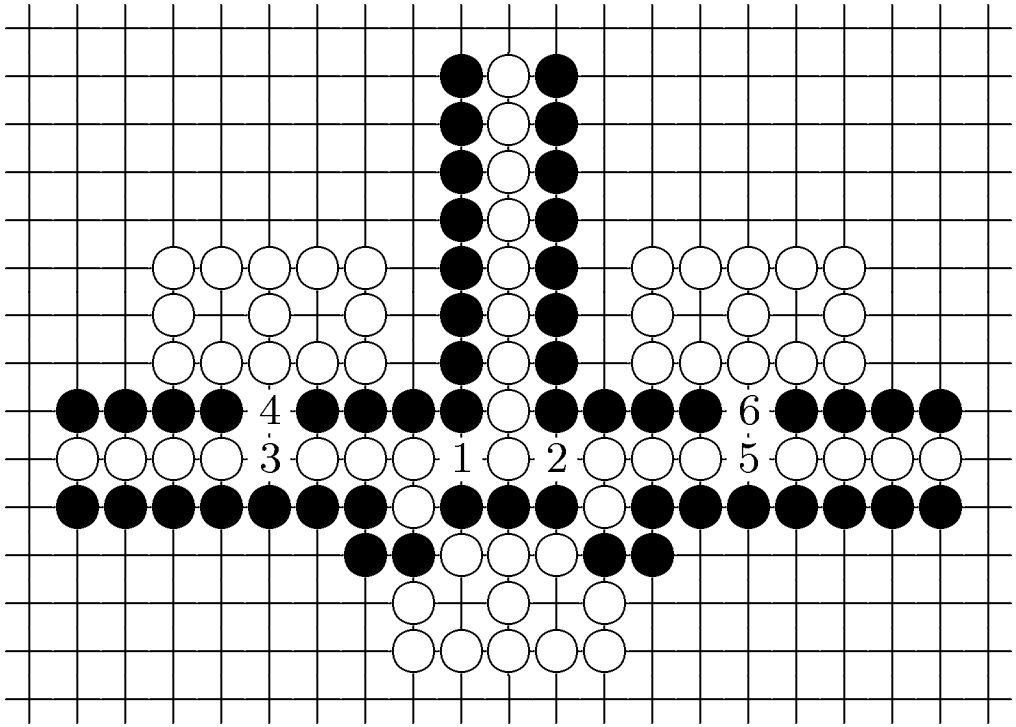
Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.

Konfigurationen in der GO-Situation

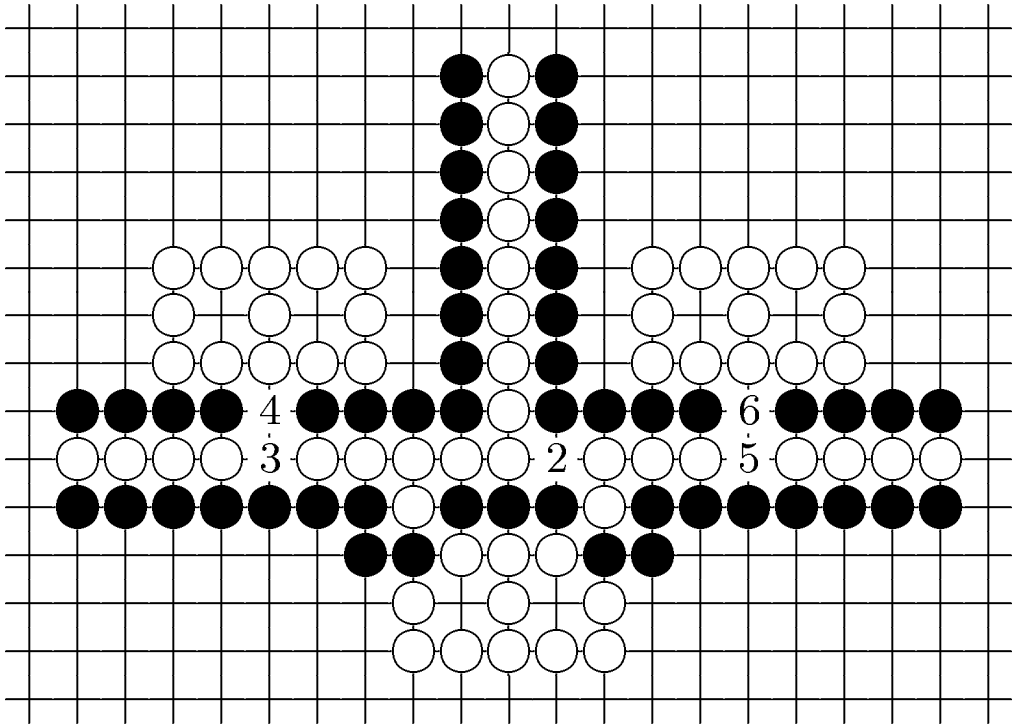
Wahlmöglichkeit Spieler 2



- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.

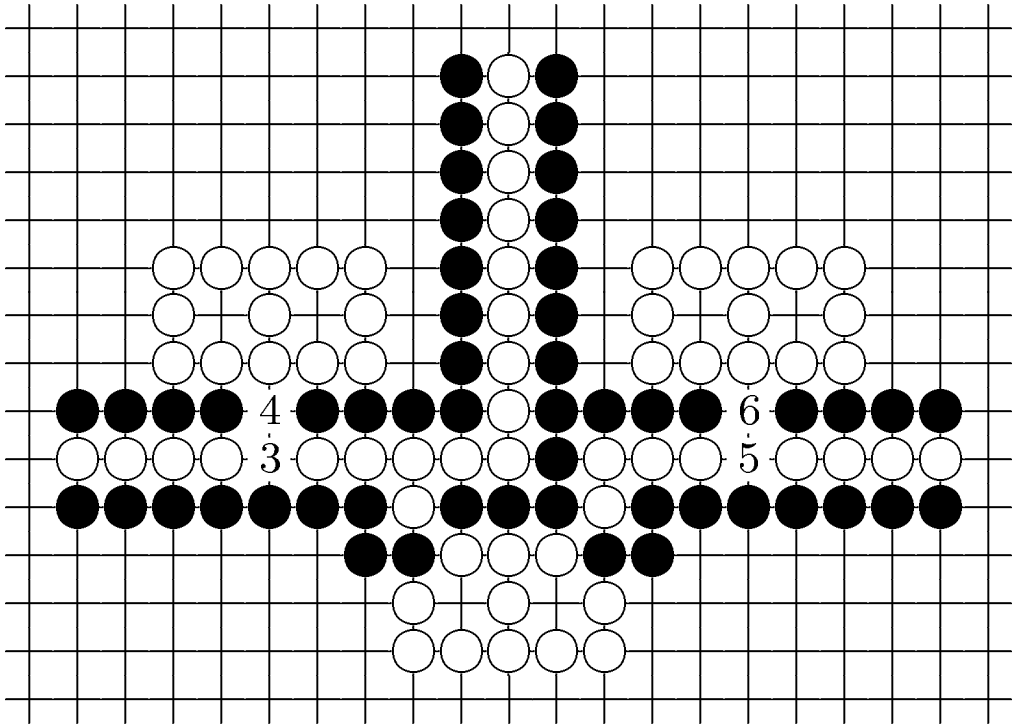
Konfigurationen in der GO-Situation

Wahlmöglichkeit Spieler 2



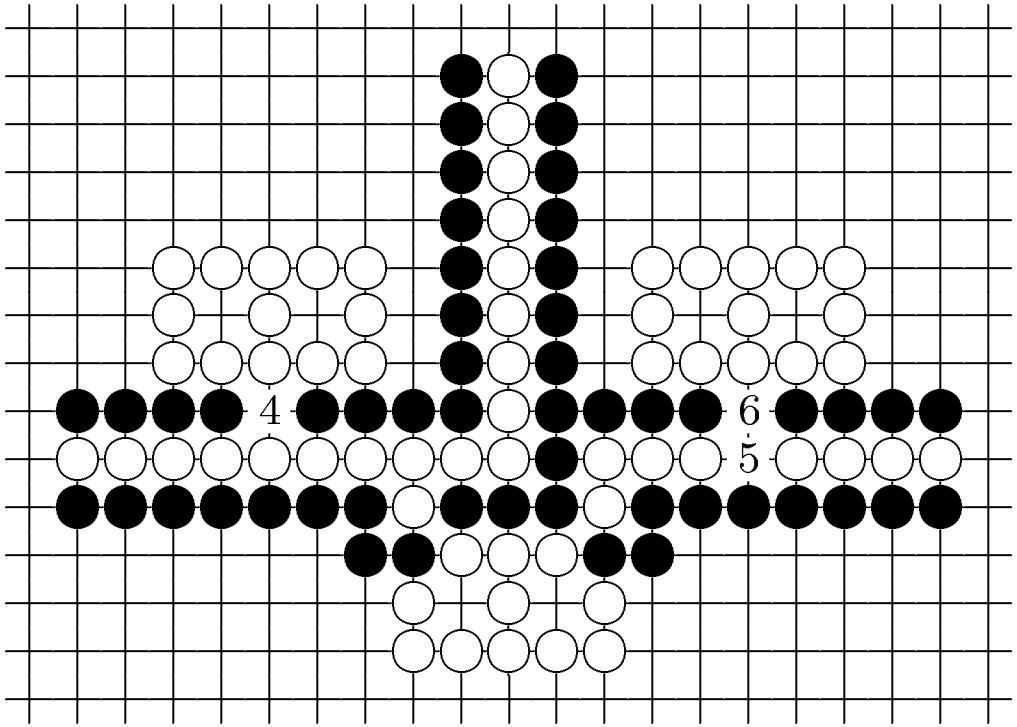
- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.

Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



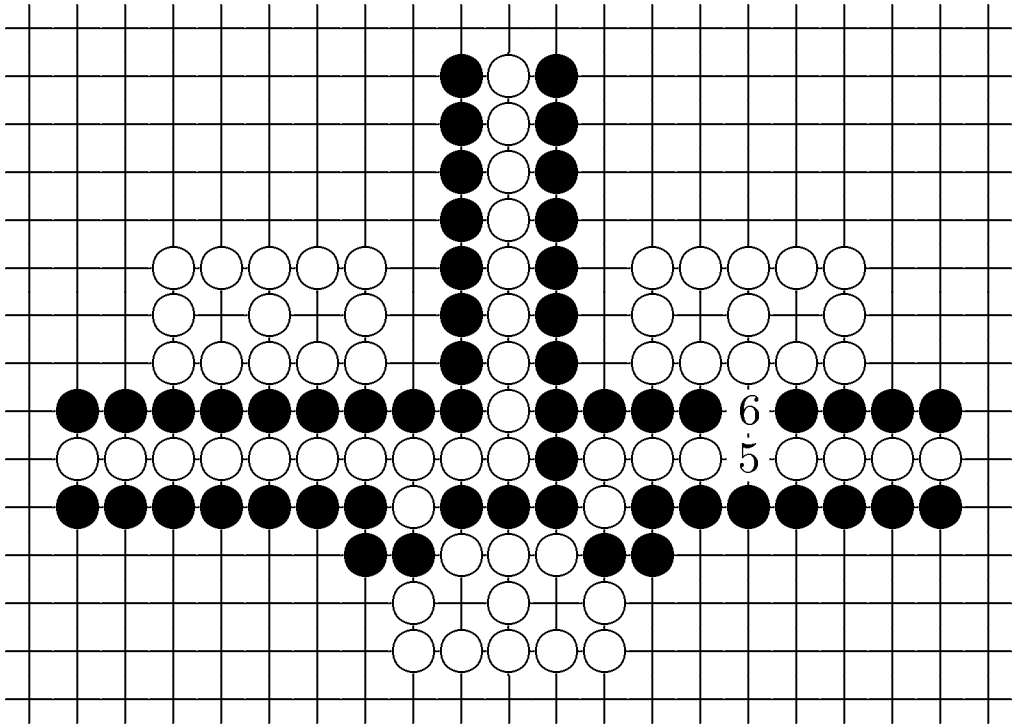
- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.
- Weiß muss im nächsten Zug auf Feld 3 antworten.

Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



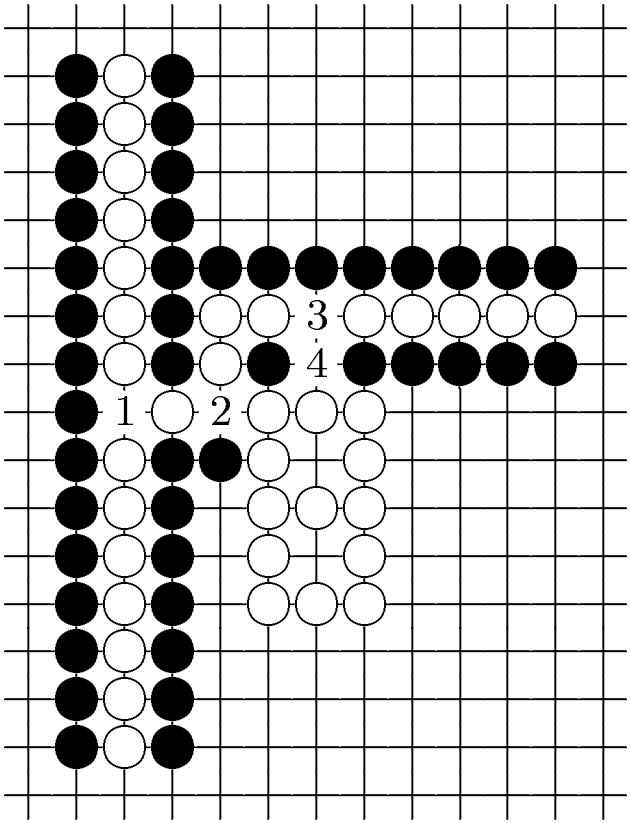
- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.
- Weiß muss im nächsten Zug auf Feld 3 antworten.
- Schwarz muss im nächsten Zug auf Feld 4 antworten.

Konfigurationen in der GO-Situation
Wahlmöglichkeit Spieler 2



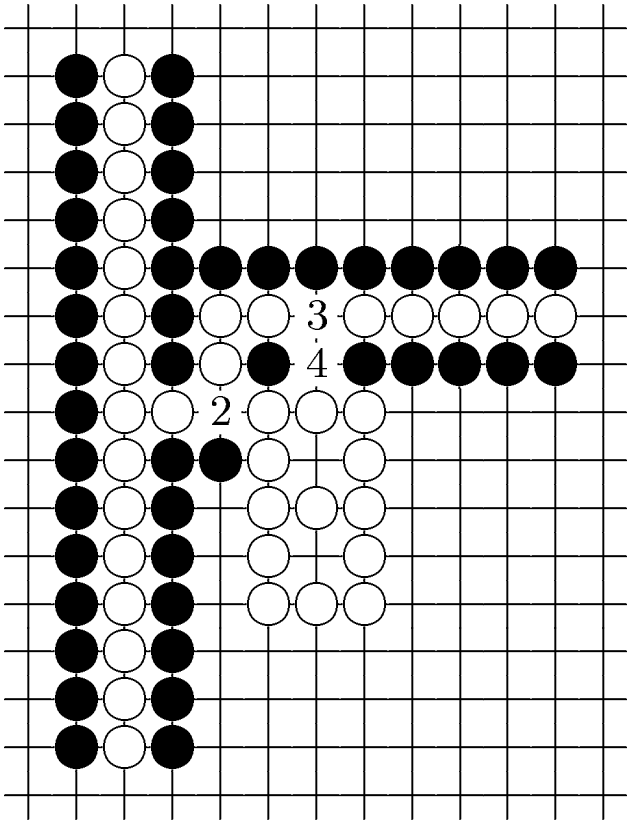
- Weiß muss im ersten Zug seinen Stein auf Feld 1 oder Feld 2 setzen.
- Schwarz muss im nächsten Zug auf Feld 2 antworten.
- Weiß muss im nächsten Zug auf Feld 3 antworten.
- Schwarz muss im nächsten Zug auf Feld 4 antworten.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



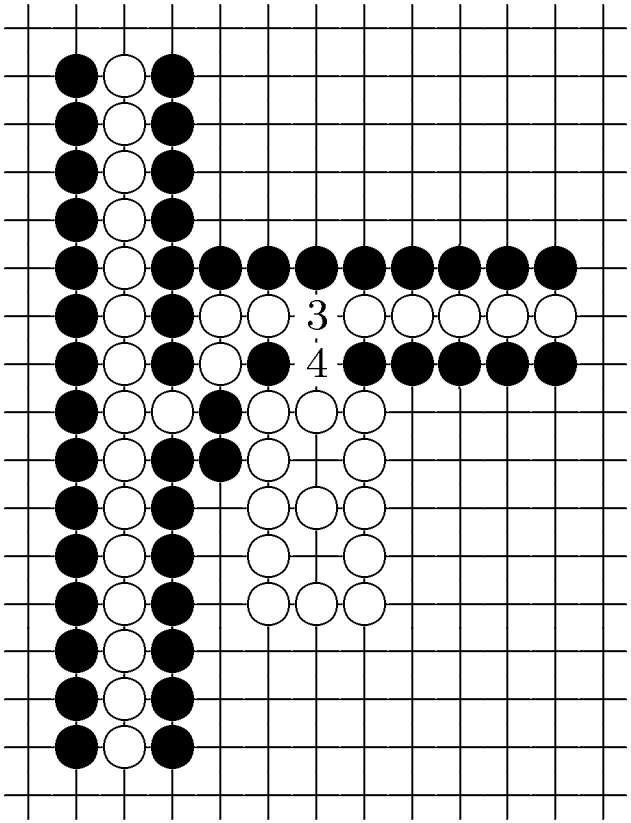
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



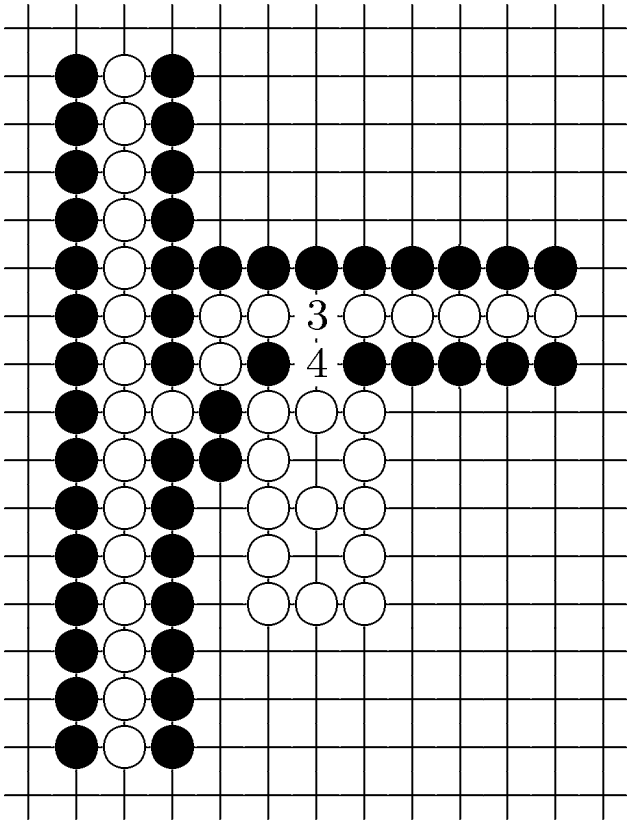
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



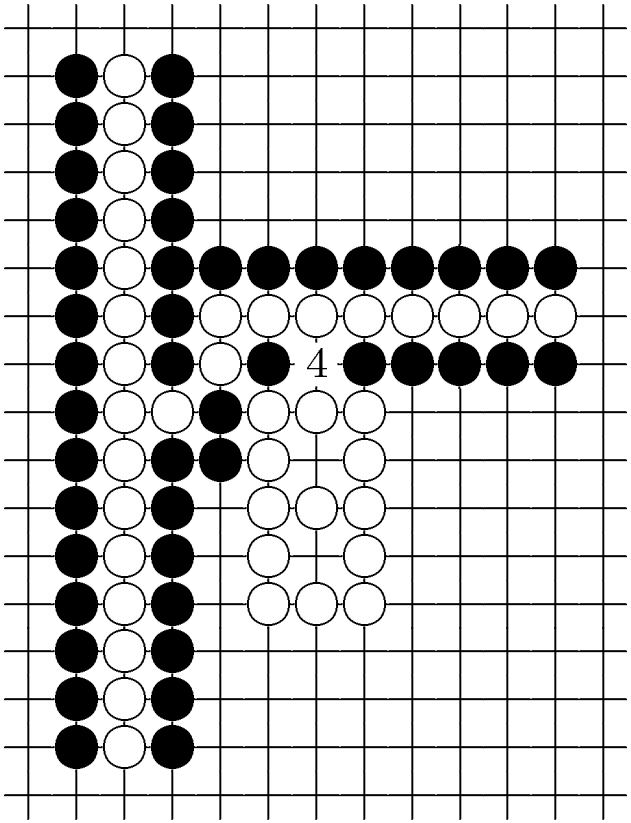
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



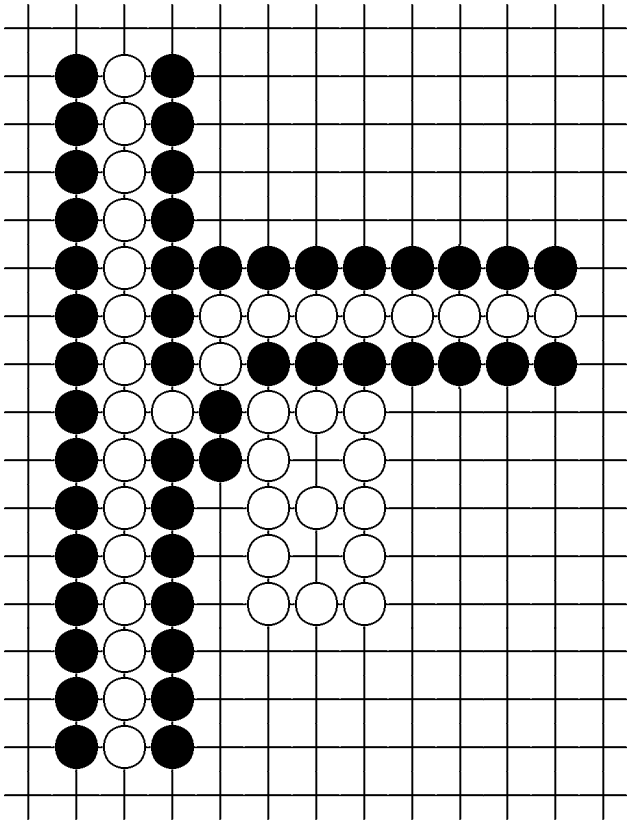
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.
- Wird dieselbe Test-Konfiguration noch einmal von rechts betreten, gewinnt Schwarz.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



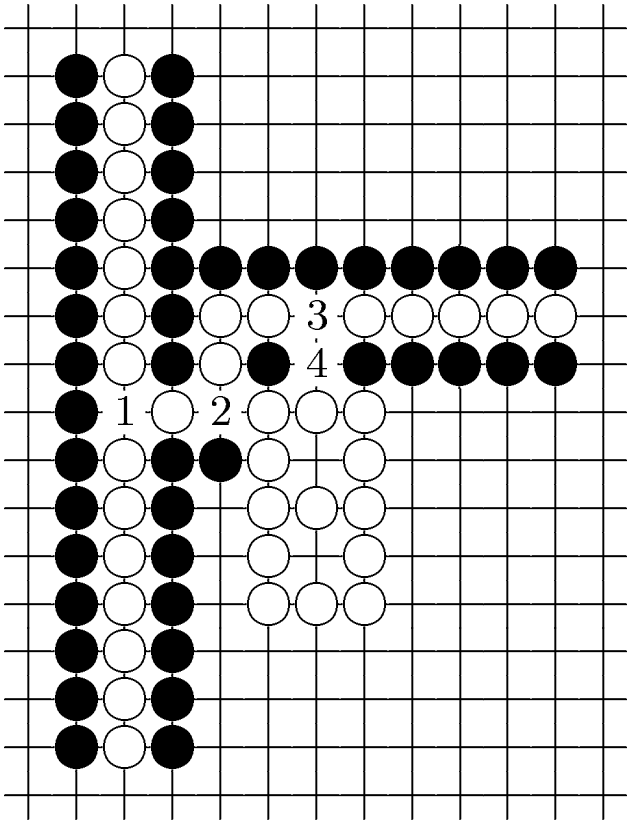
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.
- Wird dieselbe Test-Konfiguration noch einmal von rechts betreten, gewinnt Schwarz.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



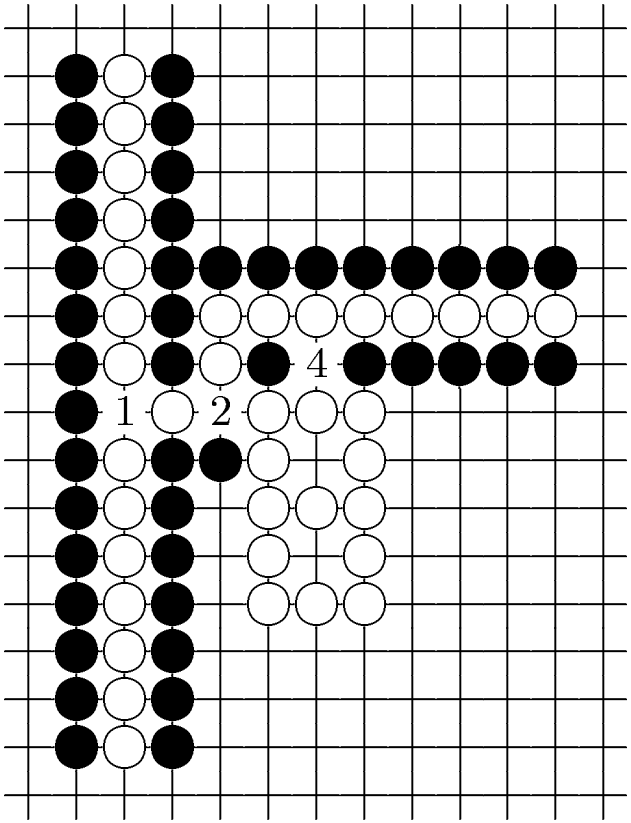
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.
- Wird dieselbe Test-Konfiguration noch einmal von rechts betreten, gewinnt Schwarz.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



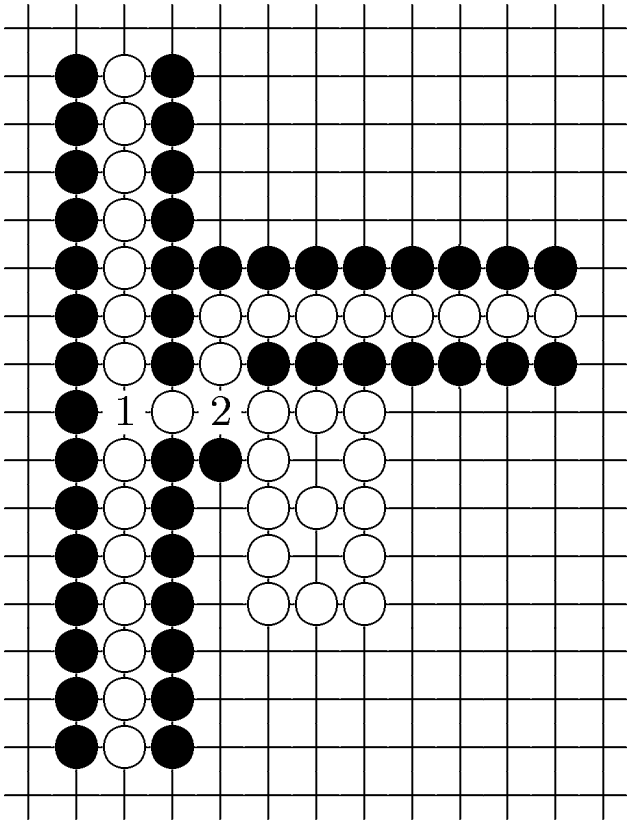
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.
- Wird dieselbe Test-Konfiguration noch einmal von rechts betreten, gewinnt Schwarz.
- Wird eine Test-Konfiguration beim ersten Mal von rechts betreten, gewinnt Weiß.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



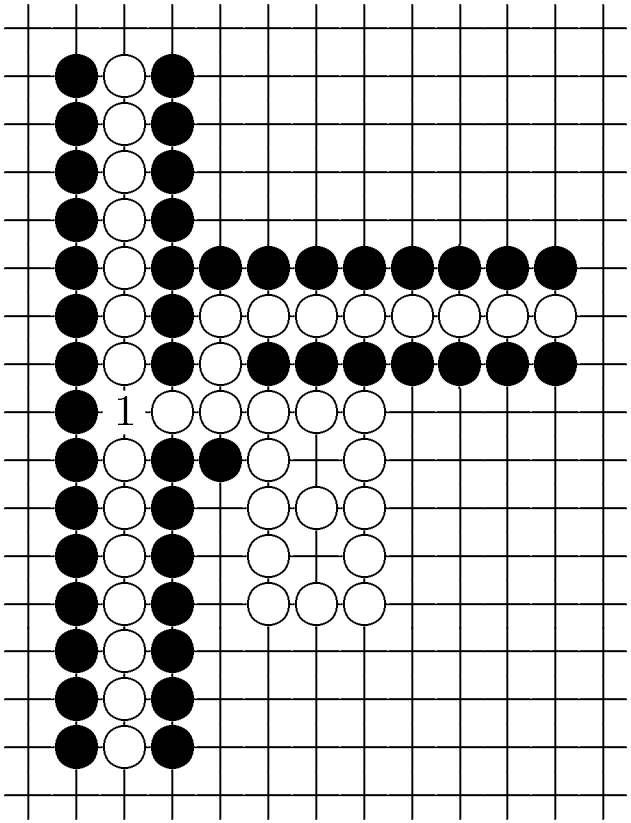
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.
- Wird dieselbe Test-Konfiguration noch einmal von rechts betreten, gewinnt Schwarz.
- Wird eine Test-Konfiguration beim ersten Mal von rechts betreten, gewinnt Weiß.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke



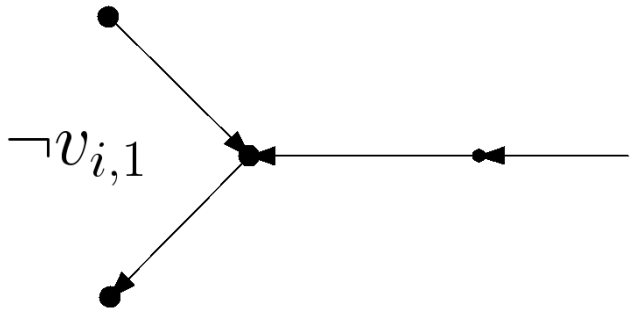
- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.
- Wird dieselbe Test-Konfiguration noch einmal von rechts betreten, gewinnt Schwarz.
- Wird eine Test-Konfiguration beim ersten Mal von rechts betreten, gewinnt Weiß.

Konfigurationen in der GO-Situation
Konfiguration der Test-Ecke

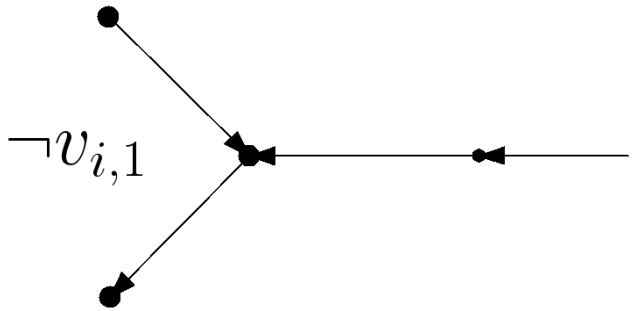


- Wird eine Test-Konfiguration zum ersten Mal von oben betreten, dann wird sie unten verlassen und ein weißer Stein ist an Feld 1, ein schwarzer an Feld 2 gesetzt.
- Wird dieselbe Test-Konfiguration noch einmal von rechts betreten, gewinnt Schwarz.
- Wird eine Test-Konfiguration beim ersten Mal von rechts betreten, gewinnt Weiß.

- Zurück zum Anfang: TQBF und GG.

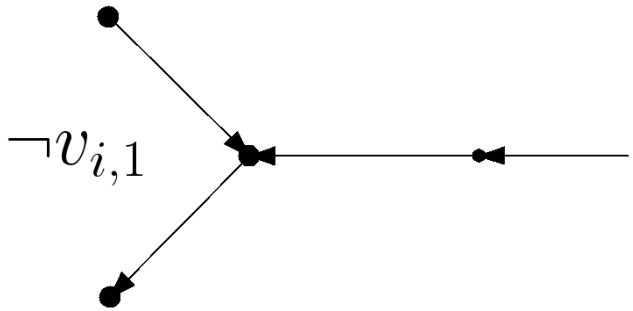


- Zurück zum Anfang: TQBF und GG.



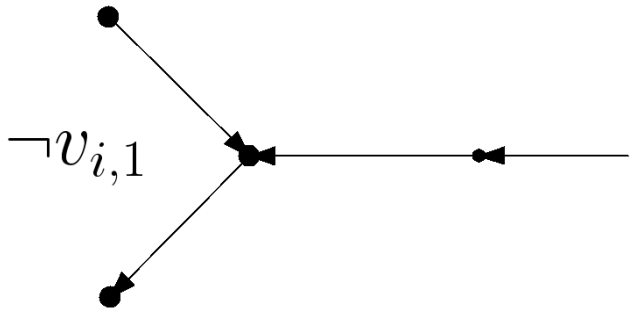
- Wurde eine Test-Konfiguration bereits durchlaufen und wird dann noch einmal von rechts betreten, so entspricht das dem Auswählen einer *erfüllenden* Variable.
  $\Rightarrow$ Schwarz (Spieler 1) gewinnt!

- Zurück zum Anfang: TQBF und GG.



- Wurde eine Test-Konfiguration bereits durchlaufen und wird dann noch einmal von rechts betreten, so entspricht das dem Auswählen einer *erfüllenden* Variable.
  $\Rightarrow$ Schwarz (Spieler 1) gewinnt!
- Wird die Konfiguration einer Testecke das erste mal von rechts betreten, so entspricht das dem Auswählen einer *nicht erfüllenden* Variable.
  $\Rightarrow$ Weiß (Spieler 2) gewinnt!

- Zurück zum Anfang: TQBF und GG.



- Wurde eine Test-Konfiguration bereits durchlaufen und wird dann noch einmal von rechts betreten, so entspricht das dem Auswählen einer *erfüllenden* Variable.
  $\Rightarrow$ Schwarz (Spieler 1) gewinnt!
- Wird die Konfiguration einer Testecke das erste mal von rechts betreten, so entspricht das dem Auswählen einer *nicht erfüllenden* Variable.
  $\Rightarrow$ Weiß (Spieler 2) gewinnt!
- Damit ist der Beweis erbracht. $\square$

Vielen Dank!

Quellen, Literatur


D. Lichtenstein, M. Sipser *GO is Polynomial-Space Hard*
Journal of the ACM, Vol 27, No 2, April 1980, pp 393-401


A. R. Meyer, L. J. Stockmeyer *Word problems requiring exponential time*
Preliminary Report Proc 5th Annual ACM Symp Theory Cmptg, Austin, Texas, 1973, pp 1-9


D. Lichtenstein *Planar formulae and their uses*
SIAM Journal of Computing, Vol 11, No 2, May 1982, pp 329-343


The Japanese Rules Of Go
`http://www.cs.cmu.edu/~wjh/go/rules/Japanese.html`