Motion Planning

Elmar Langetepe
University of Bonn

Voronoi Diagramm von Liniensegmenten

Voronoi Diagramm von Liniensegmenten

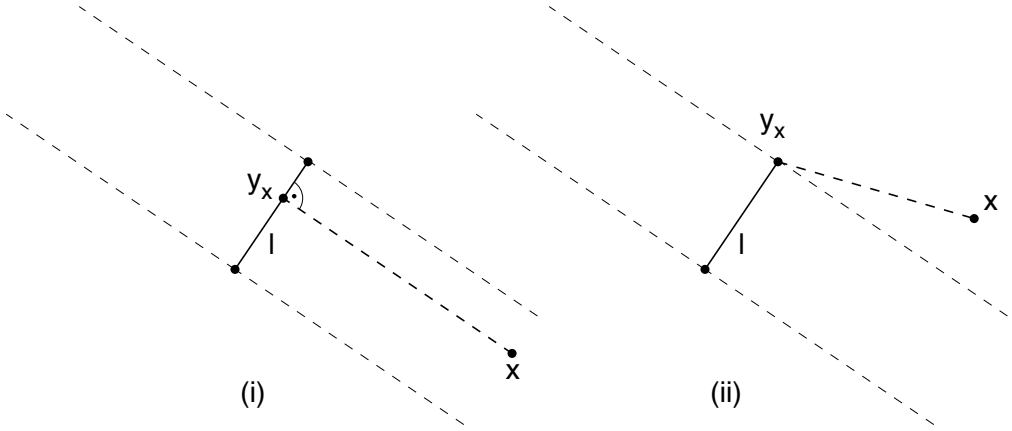
- Jetzt alle Objekte Punkte oder Liniensegmente

Voronoi Diagramm von Liniensegmenten

- Jetzt alle Objekte Punkte oder Liniensegmente
- Bisektor zwischen Punkt und Segment

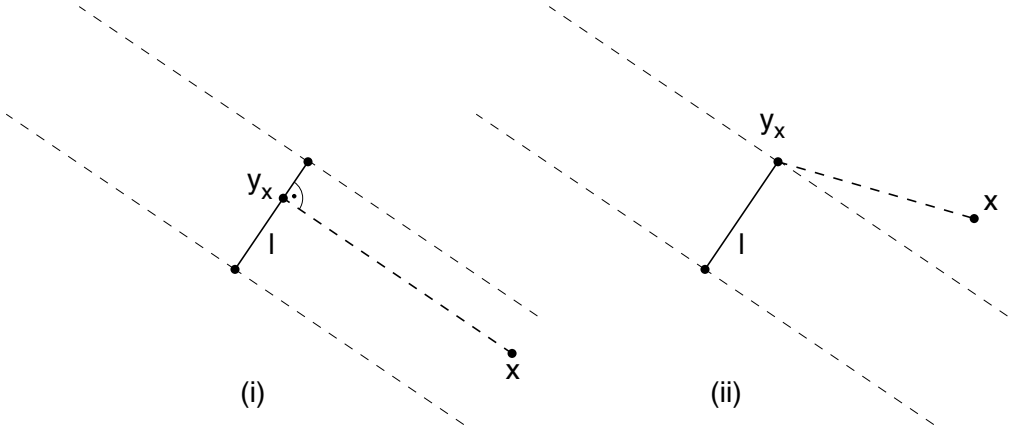
Voronoi Diagramm von Liniensegmenten

- Jetzt alle Objekte Punkte oder Liniensegmente
- Bisektor zwischen Punkt und Segment
- Abstand eines Punktes x zu Segment l, Streifen



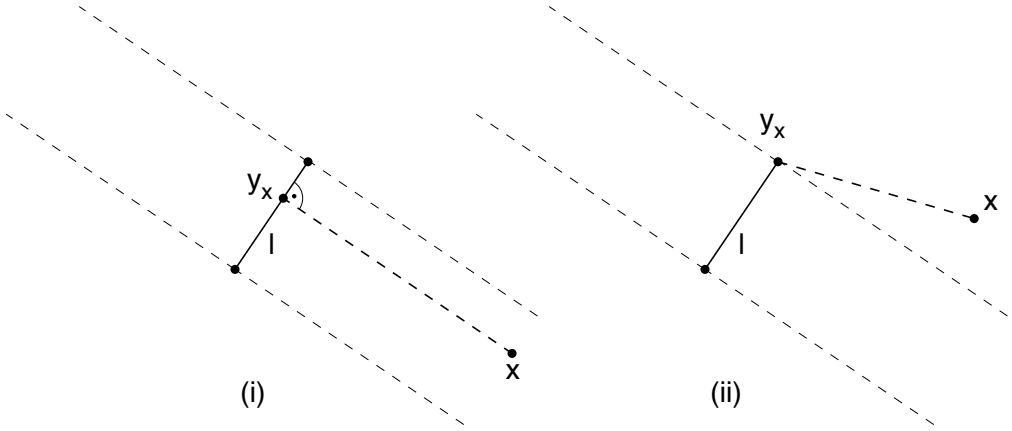
Voronoi Diagramm von Liniensegmenten

- Jetzt alle Objekte Punkte oder Liniensegmente
- Bisektor zwischen Punkt und Segment
- Abstand eines Punktes x zu Segment l, Streifen
- Bisektor zwischen zwei Segmenten l_1 und l_2



Voronoi Diagramm von Liniensegmenten

- Jetzt alle Objekte Punkte oder Liniensegmente
- Bisektor zwischen Punkt und Segment
- Abstand eines Punktes x zu Segment l, Streifen
- Bisektor zwischen zwei Segmenten l_1 und l_2
- $B(l_1, l_2) = \{x \in \mathbb{R}^2; |xl_1| = xl_2|\}$



Bisektor von Segmenten

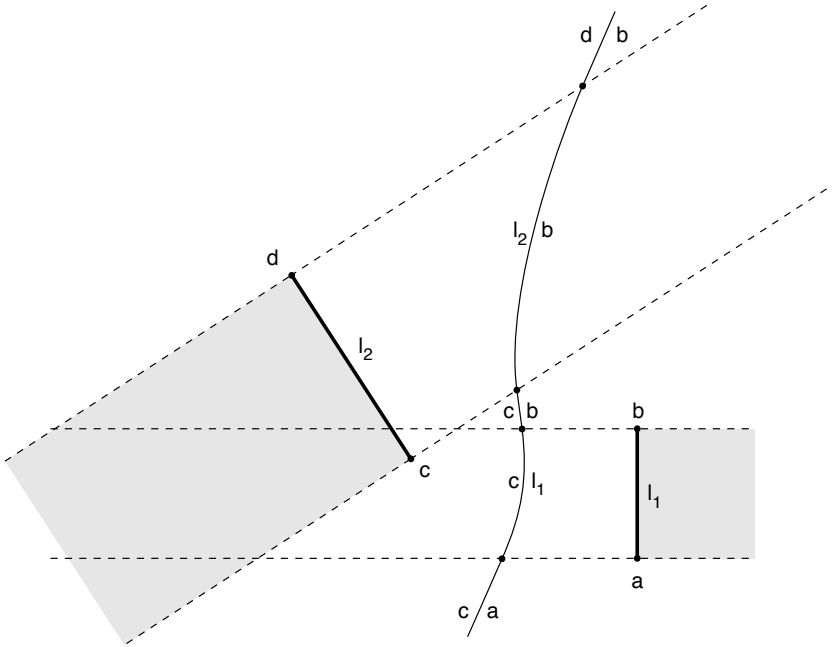
Bisektor von Segmenten

Lemma 5.24 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden.

Bisektor von Segmenten

Lemma 5.24 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden.

Verantwortungsbereiche der Streifen, Lage zueinander

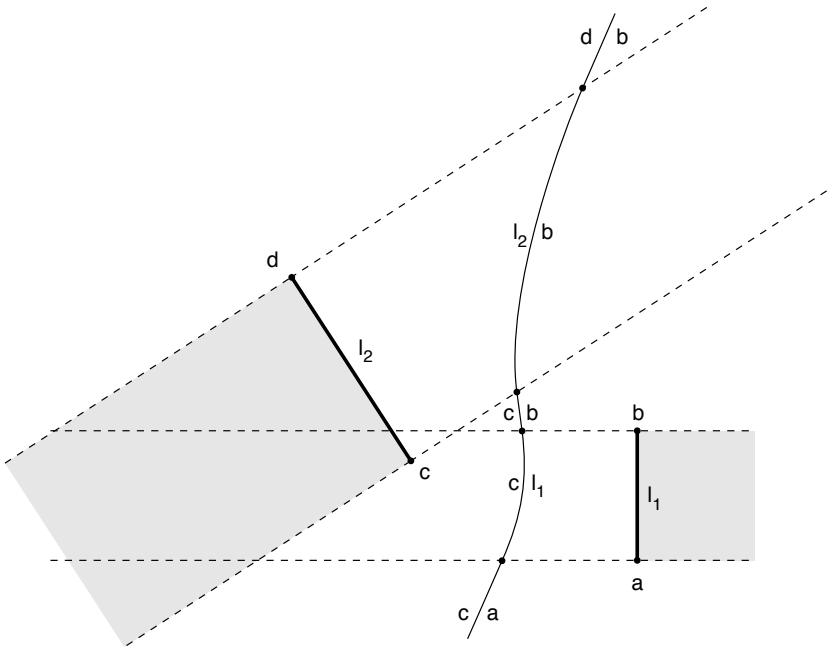


Bisektor von Segmenten

Lemma 5.24 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden.

Verantwortungsbereiche der Streifen, Lage zueinander

1. l_1 Punkt, l_2 Punkt:
   Bisektorstück Gerade

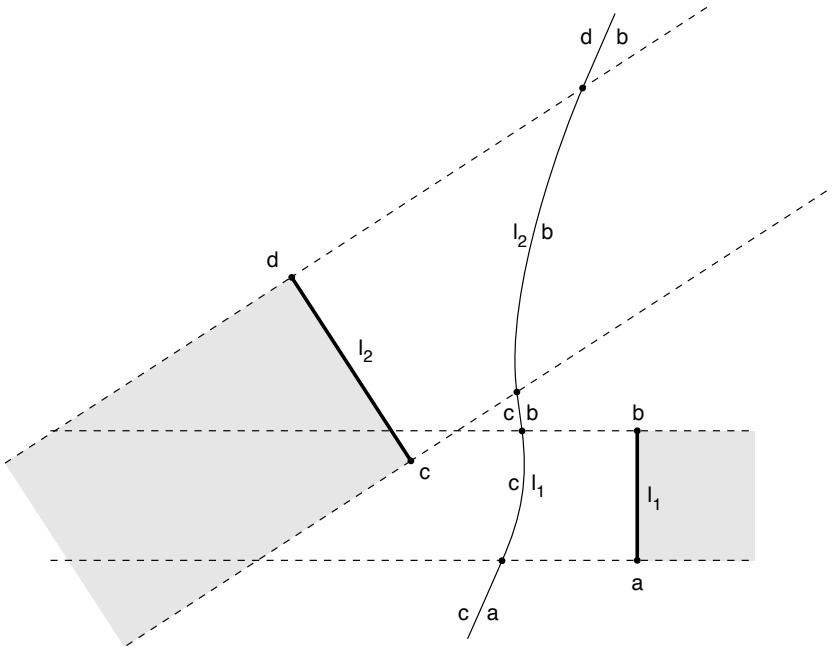


Bisektor von Segmenten

Lemma 5.24 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden.

Verantwortungsbereiche der Streifen, Lage zueinander

1. l_1 Punkt, l_2 Punkt:
   Bisektorstück Gerade
2. l_1 Segment, l_2 Punkt:
   Bisektorstück Parabel

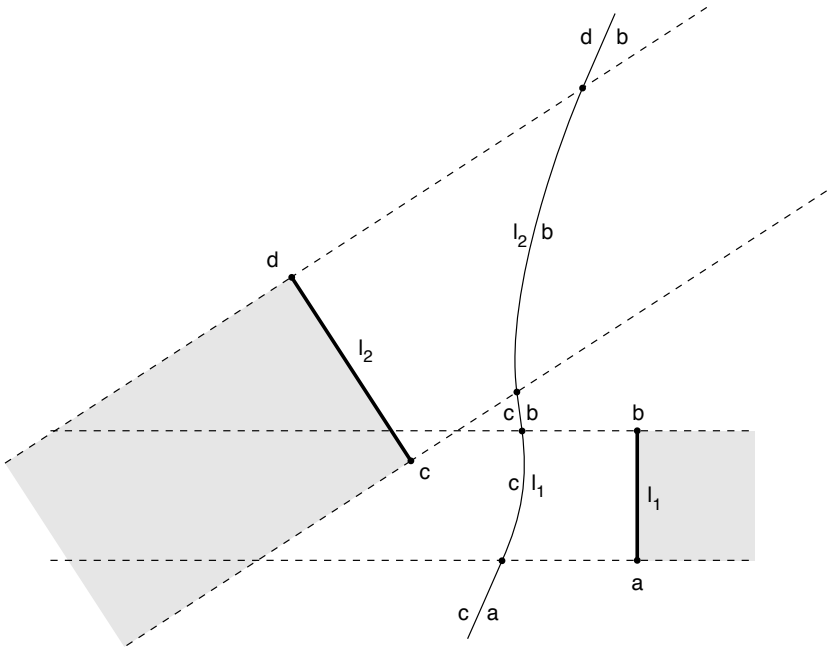


Bisektor von Segmenten

Lemma 5.24 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden.

Verantwortungsbereiche der Streifen, Lage zueinander

1. l_1 Punkt, l_2 Punkt:
   Bisektorstück Gerade
2. l_1 Segment, l_2 Punkt:
   Bisektorstück Parabel
3. l_1 Segment, l_2 Segment:
   Bisektorstück Gerade



Regionen sind sternförmig

Regionen sind sternförmig

Menge von Liniensegmenten S, Voronoi-Region von einem Liniensegment VR(l, S)!

Regionen sind sternförmig

Menge von Liniensegmenten S, Voronoi-Region von einem Liniensegment VR(l, S)!

Lemma 5.25 Sei S eine Menge von Liniensegmenten und $l \in S$. Für jeden Punkt x in der Voronoi-Region VR(l, S) gilt: Das Liniensegment xy_x zwischen x und dem Punkt $y_x \in l$ der am nächsten zu x liegt, liegt in VR(l, S).

Regionen sind sternförmig

Menge von Liniensegmenten S, Voronoi-Region von einem Liniensegment VR(l, S)!

Lemma 5.25 Sei S eine Menge von Liniensegmenten und $l \in S$. Für jeden Punkt x in der Voronoi-Region VR(l, S) gilt: Das Liniensegment xy_x zwischen x und dem Punkt $y_x \in l$ der am nächsten zu x liegt, liegt in VR(l, S).

Beweis:

Regionen sind sternförmig

Menge von Liniensegmenten S, Voronoi-Region von einem Liniensegment VR(l, S)!

Lemma 5.25 Sei S eine Menge von Liniensegmenten und $l \in S$. Für jeden Punkt x in der Voronoi-Region VR(l, S) gilt: Das Liniensegment xy_x zwischen x und dem Punkt $y_x \in l$ der am nächsten zu x liegt, liegt in VR(l, S).

Beweis: Widerspruch!

Regionen zusammenhängend

Regionen zusammenhängend

Korollar 5.26 Die Voronoi-Regionen von Liniensegmenten sind zusammenhängend.

Regionen zusammenhängend

Korollar 5.26 Die Voronoi-Regionen von Liniensegmenten sind zusammenhängend.

Beweis:

Regionen zusammenhängend

Korollar 5.26 Die Voronoi-Regionen von Liniensegmenten sind zusammenhängend.

Beweis: Alle Punkte der Region haben *Blickkontakt* zu l!

Bisektor: Maximal 7 Stücke!

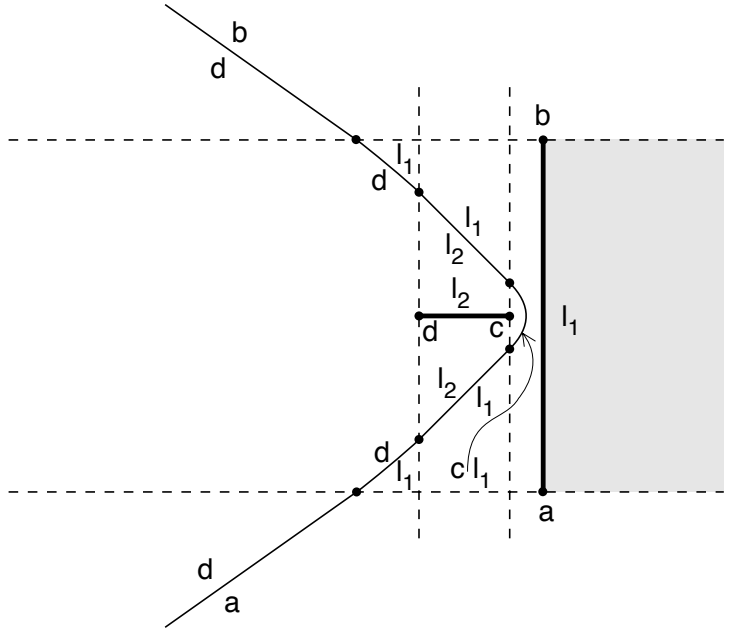
Bisektor: Maximal 7 Stücke!

Lemma 5.27 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden und besteht aus maximal 7 Stücken.

Bisektor: Maximal 7 Stücke!

Lemma 5.27 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden und besteht aus maximal 7 Stücken.

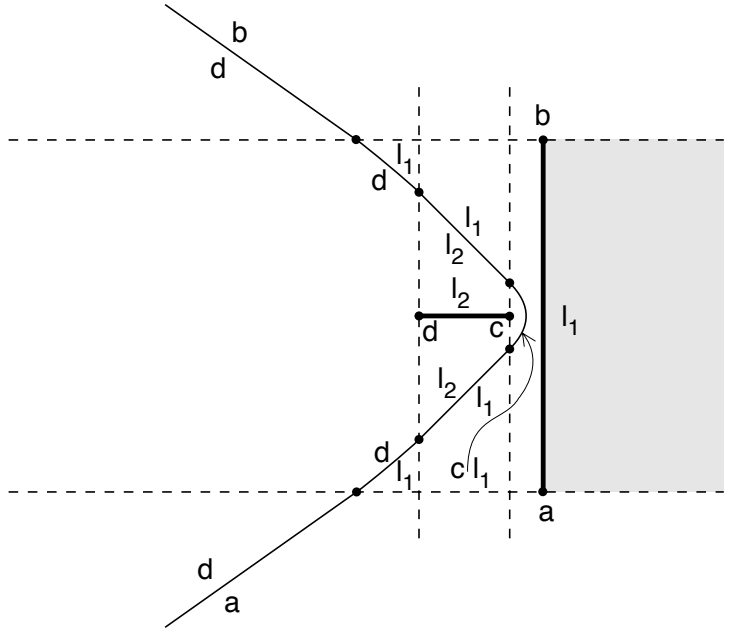
- Halbstreifen, insgesamt 8



Bisektor: Maximal 7 Stücke!

Lemma 5.27 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden und besteht aus maximal 7 Stücken.

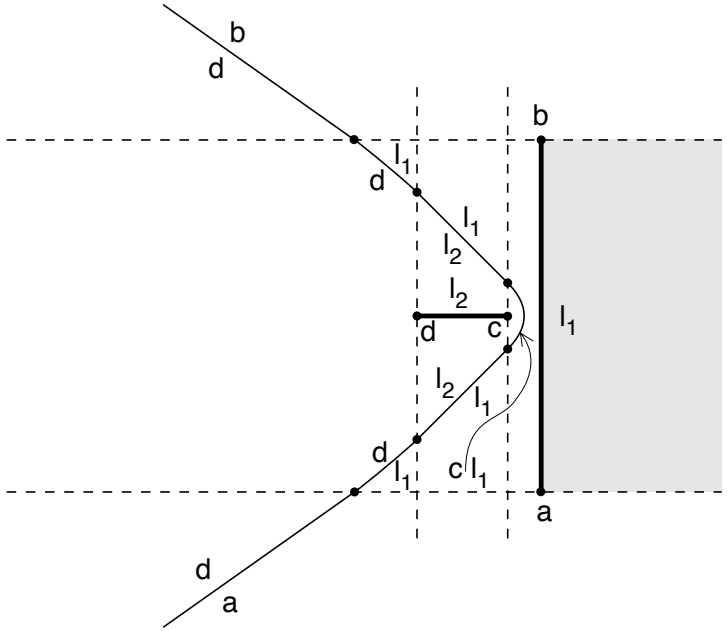
- Halbstreifen, insgesamt 8
- Einmal betreten,
  einmal verlassen, monoton



Bisektor: Maximal 7 Stücke!

Lemma 5.27 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden und besteht aus maximal 7 Stücken.

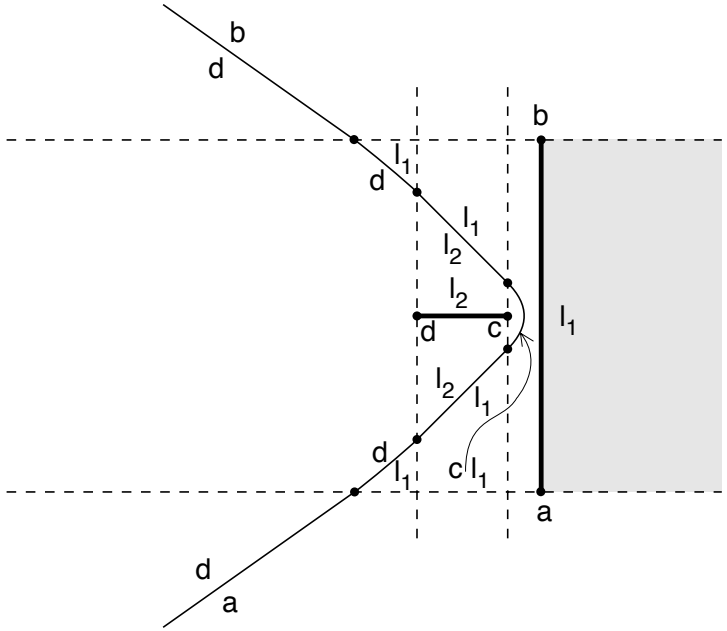
- Halbstreifen, insgesamt 8
- Einmal betreten, einmal verlassen, monoton
- Mind. Segment l_1 liegt auf konvex. Hülle



Bisektor: Maximal 7 Stücke!

Lemma 5.27 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden und besteht aus maximal 7 Stücken.

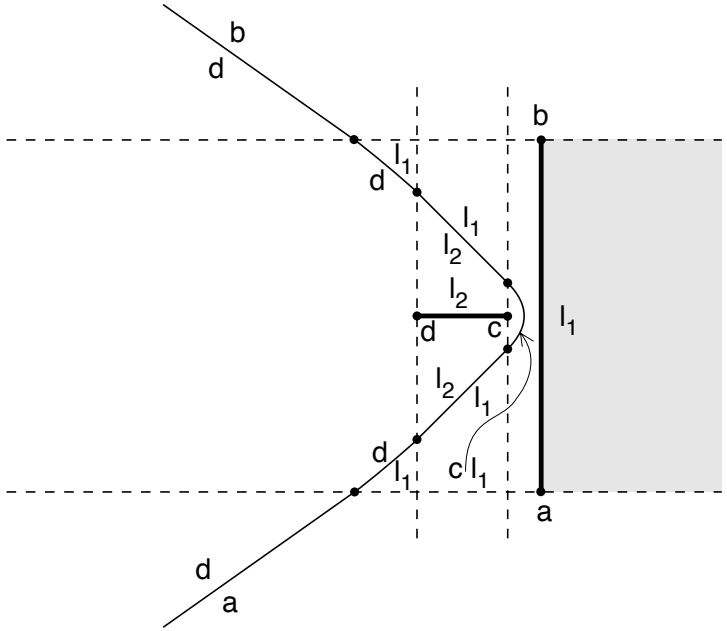
- Halbstreifen, insgesamt 8
- Einmal betreten, einmal verlassen, monoton
- Mind. Segment l_1 liegt auf konvex. Hülle
- Bisektor betritt sukzessive 3 Streifen von l_1



Bisektor: Maximal 7 Stücke!

Lemma 5.27 Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden und besteht aus maximal 7 Stücken.

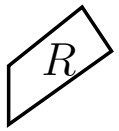
- Halbstreifen, insgesamt 8
- Einmal betreten, einmal verlassen, monoton
- Mind. Segment l_1 liegt auf konvex. Hülle
- Bisektor betritt sukzessive 3 Streifen von l_1
- max. 6 Kanten überqueren



Anwendung Bahnplanung: Kreisförmiger Agent

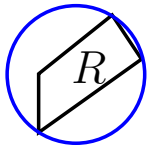
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter



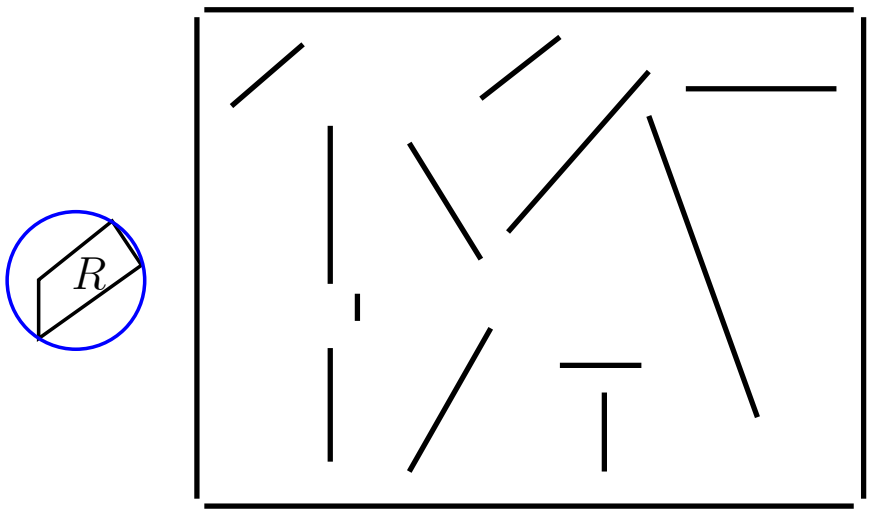
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse



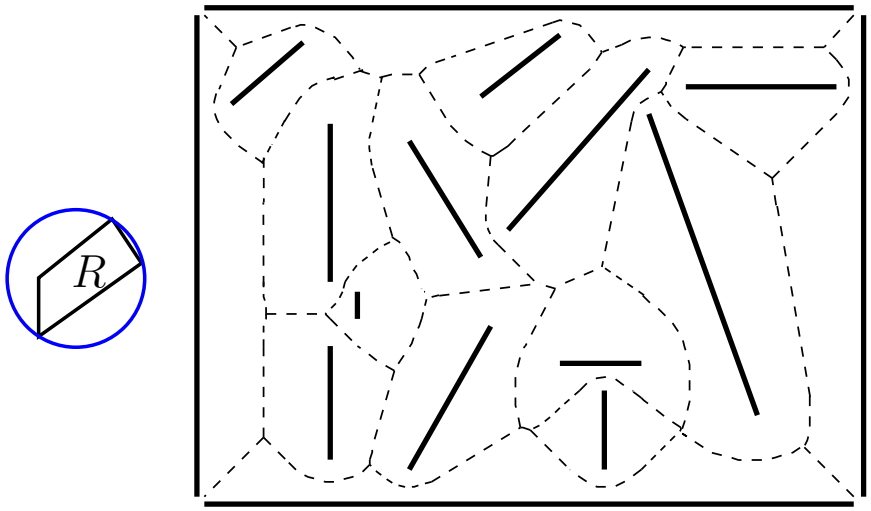
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren:



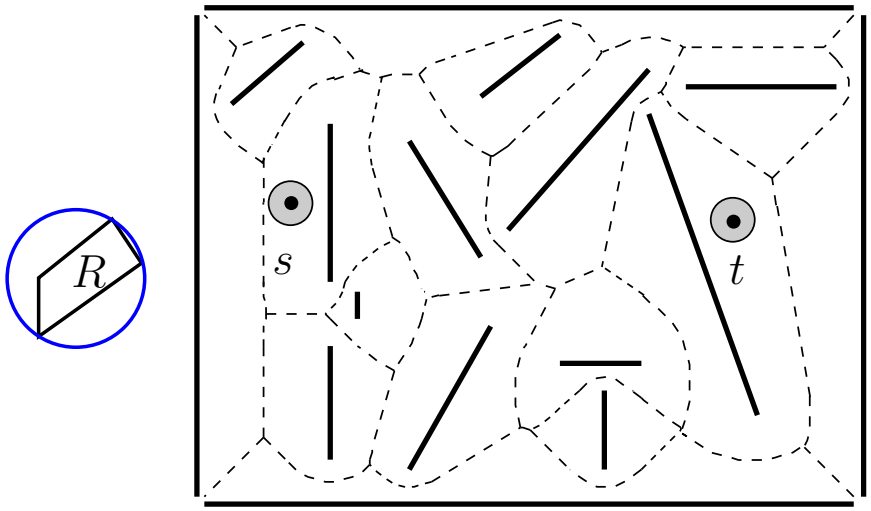
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren:



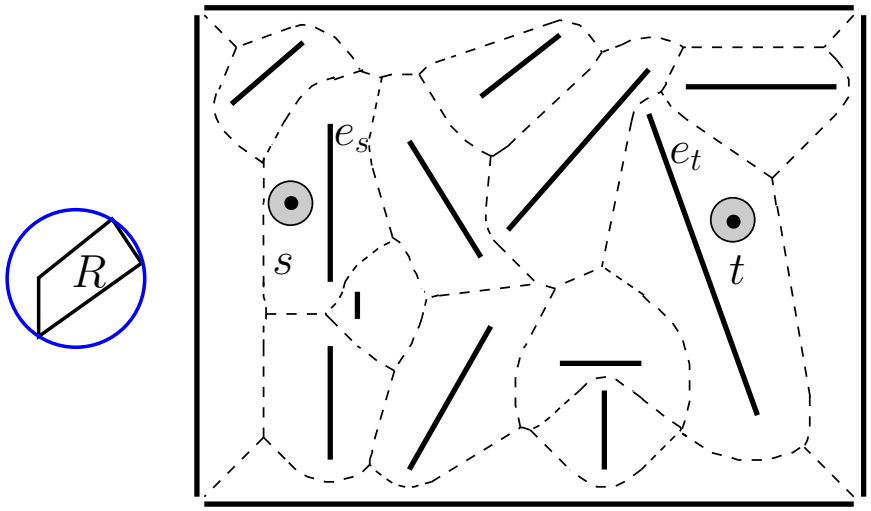
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren:



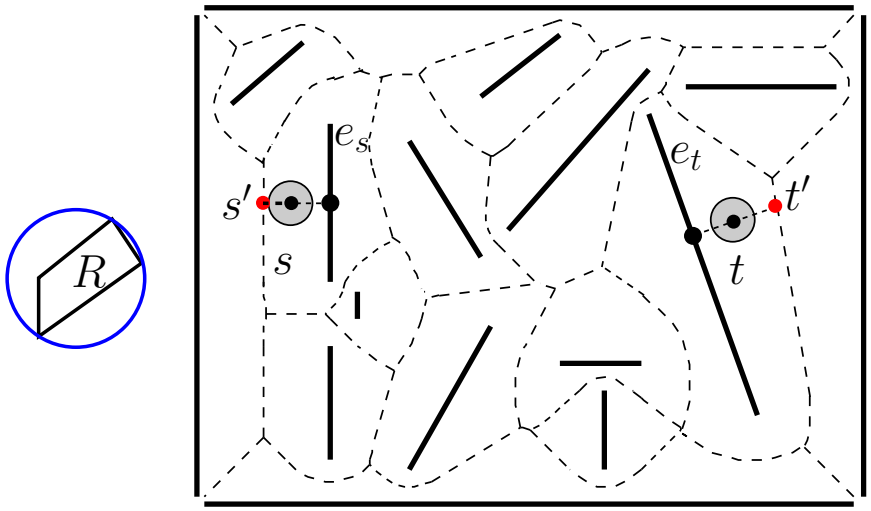
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren:



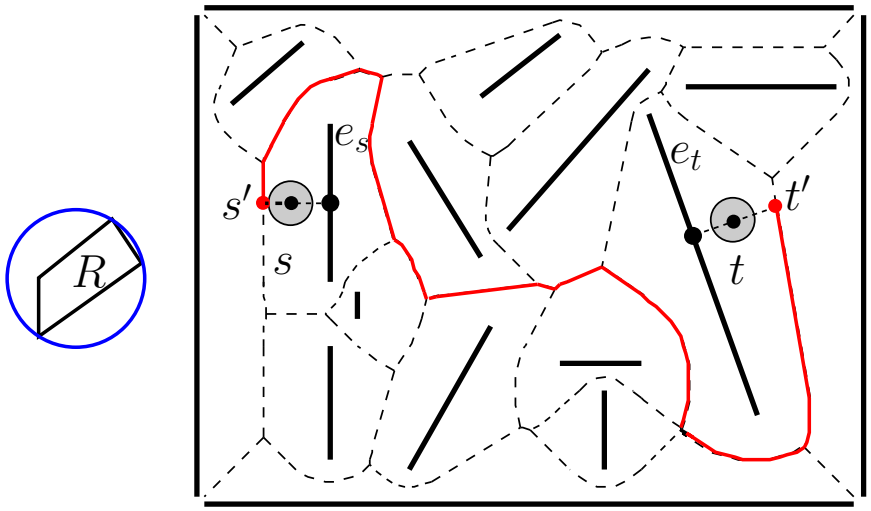
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren:



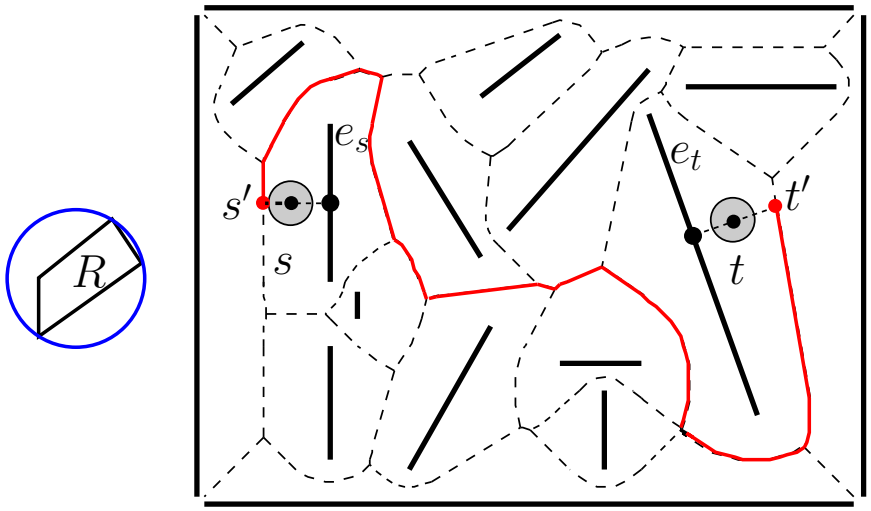
Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren:

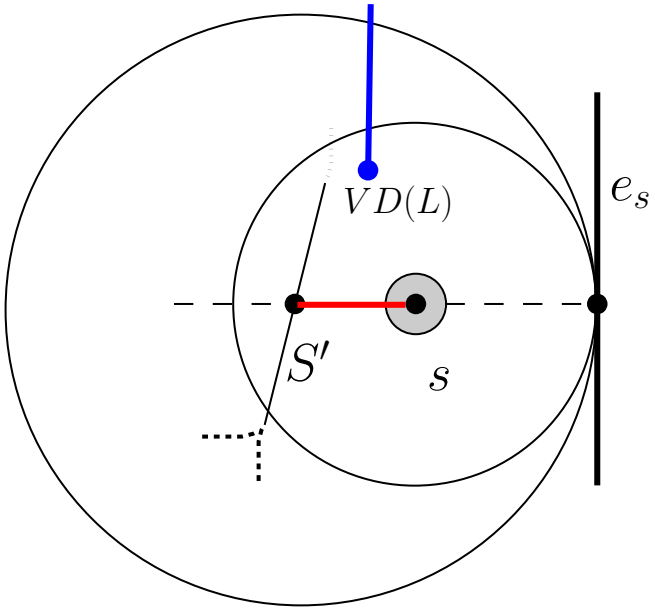


Anwendung Bahnplanung: Kreisförmiger Agent

- Verwende kleinsten Kreis um Roboter
- Voronoi Diagramm der Segmente der Hindernisse
- Weg auf Bisektoren: Möglichst großer Abstand

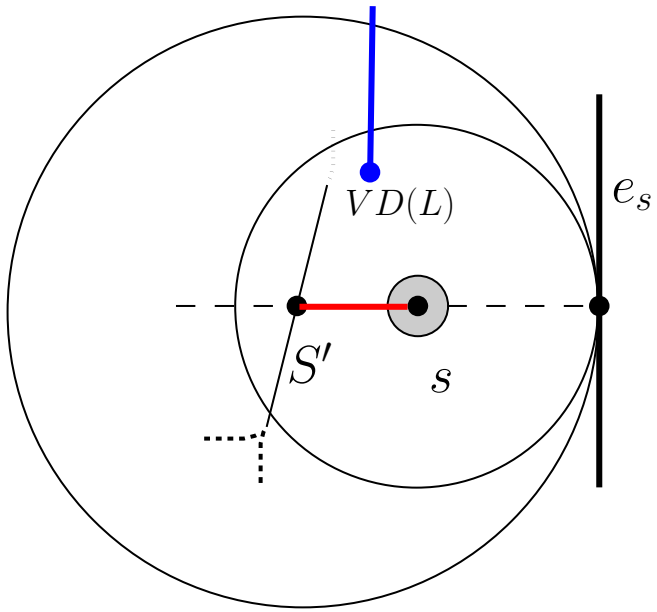


Start s' kann stets angelaufen werden



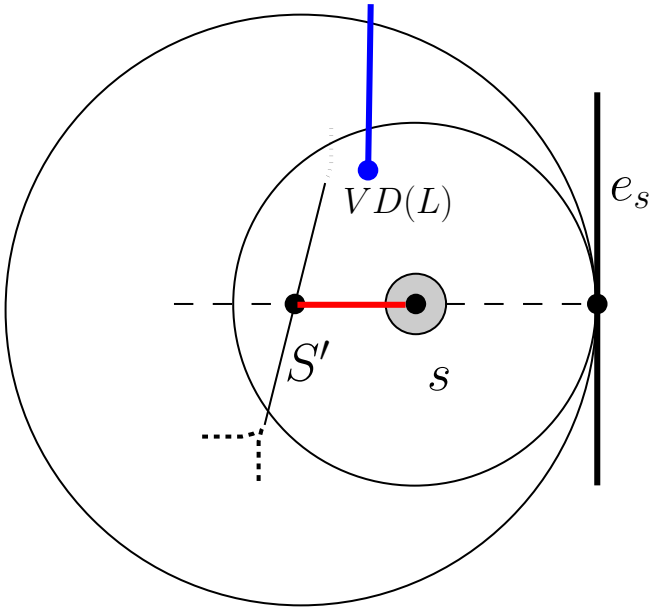
Start s' kann stets angelaufen werden

- s in Region von e_s,



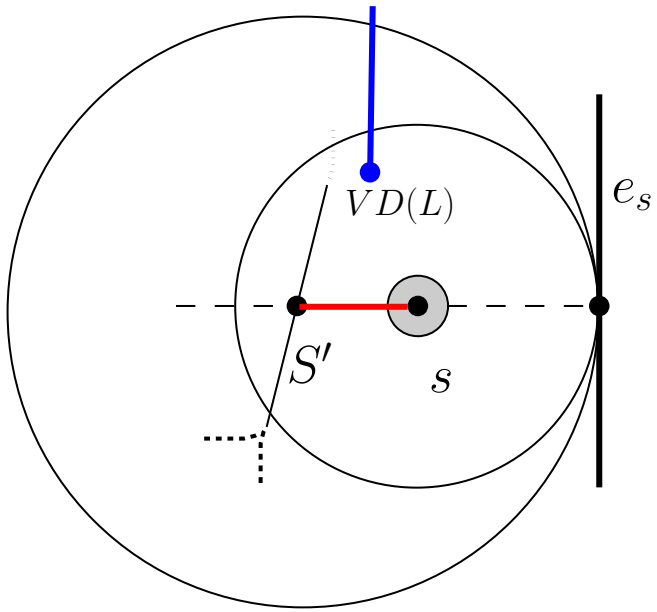
Start s' kann stets angelaufen werden

- s in Region von e_s, Kreis frei



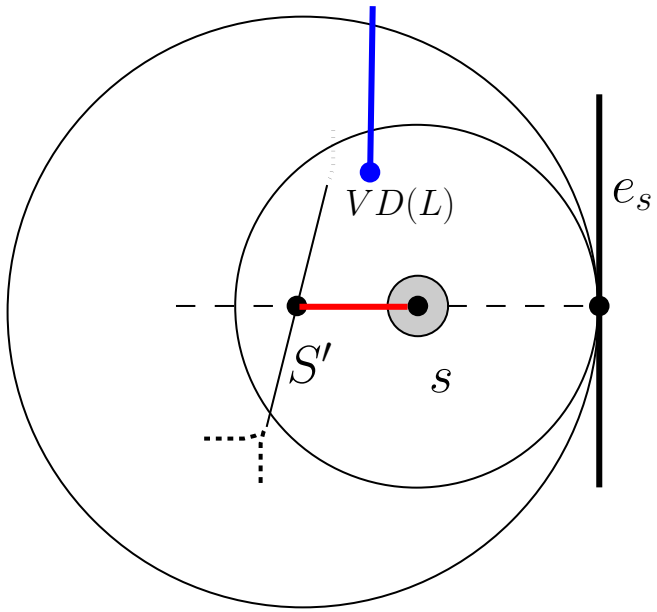
Start s' kann stets angelaufen werden

- s in Region von e_s, Kreis frei
- Kürzester Weg zu e_s, Strahl Richtung Bisektor



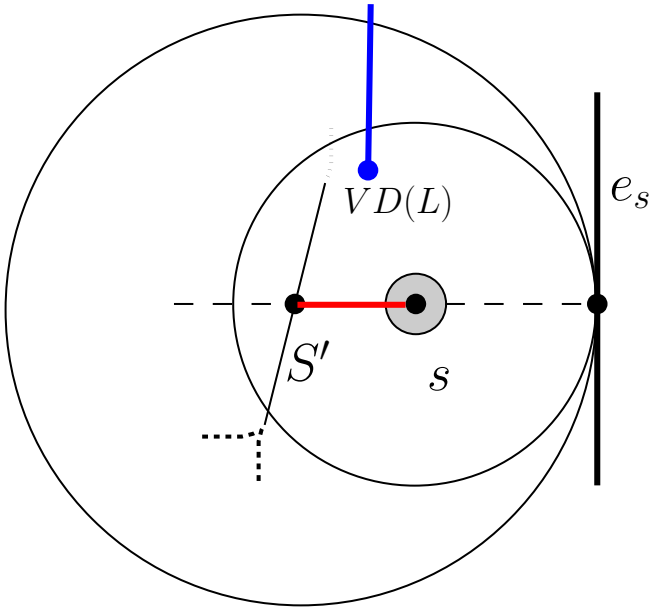
Start s' kann stets angelaufen werden

- s in Region von e_s, Kreis frei
- Kürzester Weg zu e_s, Strahl Richtung Bisektor
- Trifft Bisektor bei S',



Start s' kann stets angelaufen werden

- s in Region von e_s, Kreis frei
- Kürzester Weg zu e_s, Strahl Richtung Bisektor
- Trifft Bisektor bei S', Kreis/Weg ist frei!!



Bewegungsplanung von Agenten

Bewegungsplanung von Agenten

- Historie: Entkommen aus dem Labyrinth

Bewegungsplanung von Agenten

- Historie: Entkommen aus dem Labyrinth
- Effizientmaße: Länge des Weges/Rechenzeit

Bewegungsplanung von Agenten

- Historie: Entkommen aus dem Labyrinth
- Effizientmaße: Länge des Weges/Rechenzeit
- Offline: Alle Informationen sind vorhanden

Bewegungsplanung von Agenten

- Historie: Entkommen aus dem Labyrinth
- Effizientmaße: Länge des Weges/Rechenzeit
- Offline: Alle Informationen sind vorhanden
- Online: Nur lokale Informationen: Sicht/Tastsensor

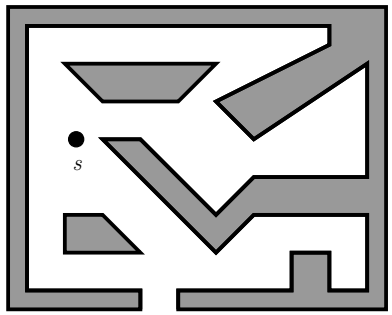
Bewegungsplanung von Agenten

- Historie: Entkommen aus dem Labyrinth
- Effizientmaße: Länge des Weges/Rechenzeit
- Offline: Alle Informationen sind vorhanden
- Online: Nur lokale Informationen: Sicht/Tastsensor
- Modell: Karte aufbauen, Umgebung merken, kein Speicher

Bewegungsplanung von Agenten

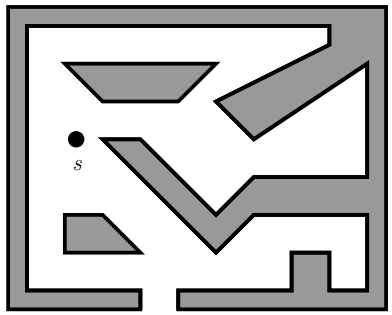
- Historie: Entkommen aus dem Labyrinth
- Effizientmaße: Länge des Weges/Rechenzeit
- Offline: Alle Informationen sind vorhanden
- Online: Nur lokale Informationen: Sicht/Tastsensor
- Modell: Karte aufbauen, Umgebung merken, kein Speicher
- Unvollständige Information

Polygonale Umgebungen



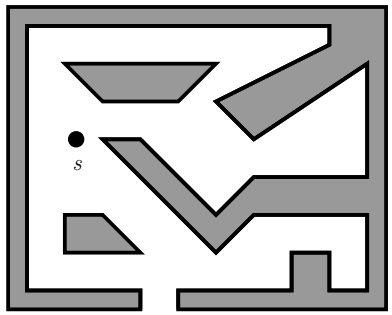
Polygonale Umgebungen

- Umgebung: Menge von disjunkten einfachen Polygonen



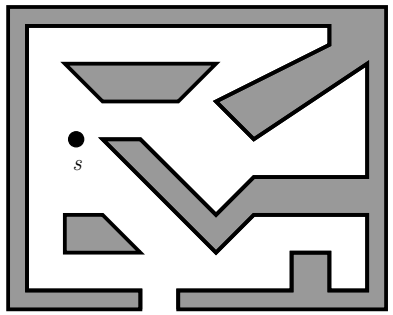
Polygonale Umgebungen

- Umgebung: Menge von disjunkten einfachen Polygonen
- Ein Randpolygon



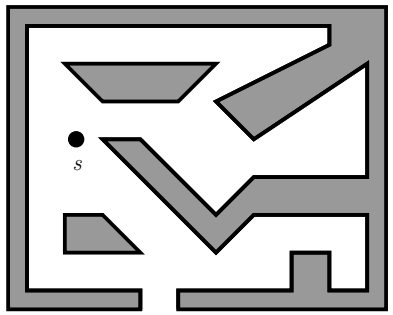
Polygonale Umgebungen

- Umgebung: Menge von disjunkten einfachen Polygonen
- Ein Randpolygon
- Verschiedene Aufgaben: Suchen eines Zielpunktes/Entkommen aus Labyrinth



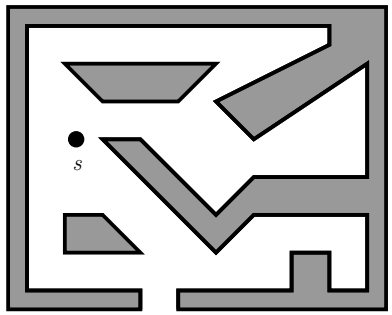
Polygonale Umgebungen

- Umgebung: Menge von disjunkten einfachen Polygonen
- Ein Randpolygon
- Verschiedene Aufgaben: Suchen eines Zielpunktes/Entkommen aus Labyrinth
- Verschiedene Sensormodelle

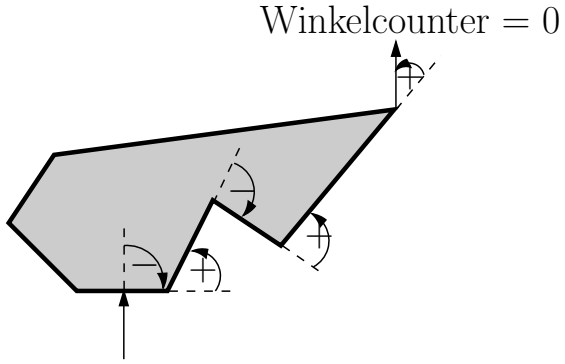


Polygonale Umgebungen

- Umgebung: Menge von disjunkten einfachen Polygonen
- Ein Randpolygon
- Verschiedene Aufgaben: Suchen eines Zielpunktes/Entkommen aus Labyrinth
- Verschiedene Sensormodelle
- Zunächst: Entkommen aus dem Labyrinth

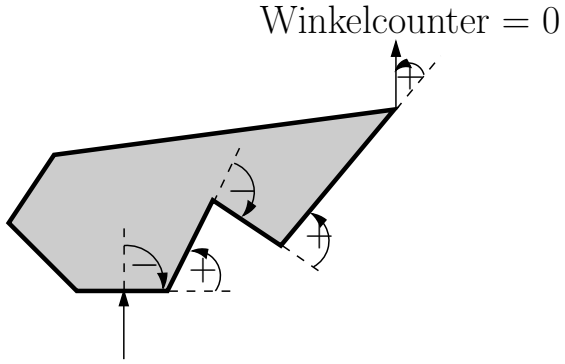


Entkommen aus dem Labyrinth: Modell



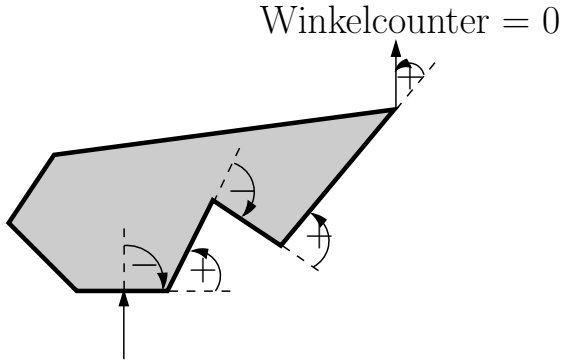
Entkommen aus dem Labyrinth: Modell

- Punktförmiger Agent/kreisförmiger Agent



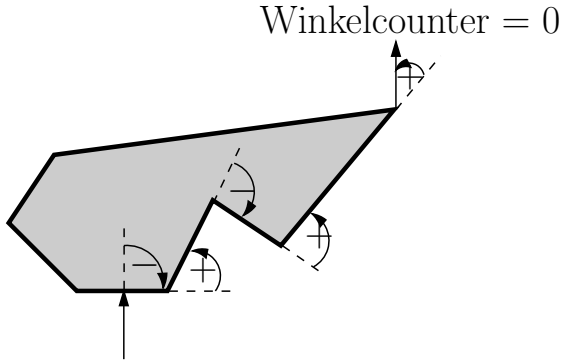
Entkommen aus dem Labyrinth: Modell

- Punktförmiger Agent/kreisförmiger Agent
- Touch Sensor



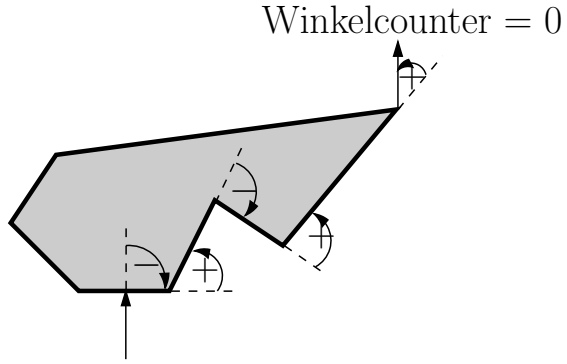
Entkommen aus dem Labyrinth: Modell

- Punktförmiger Agent/kreisförmiger Agent
- Touch Sensor
- Folge einer Wand



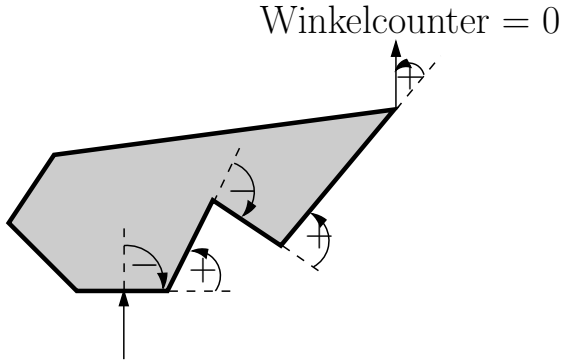
Entkommen aus dem Labyrinth: Modell

- Punktförmiger Agent/kreisförmiger Agent
- Touch Sensor
- Folge einer Wand
- Folge einer Richtung (exakt)



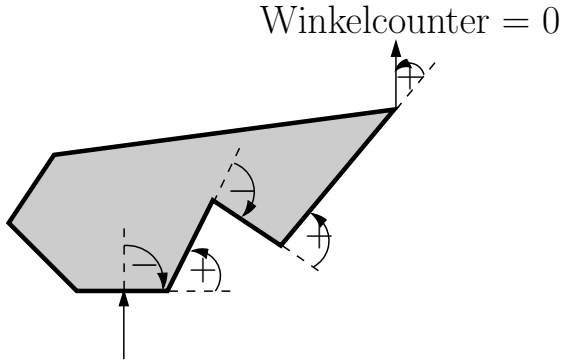
Entkommen aus dem Labyrinth: Modell

- Punktförmiger Agent/kreisförmiger Agent
- Touch Sensor
- Folge einer Wand
- Folge einer Richtung (exakt)
- Drehwinkel-Zähler

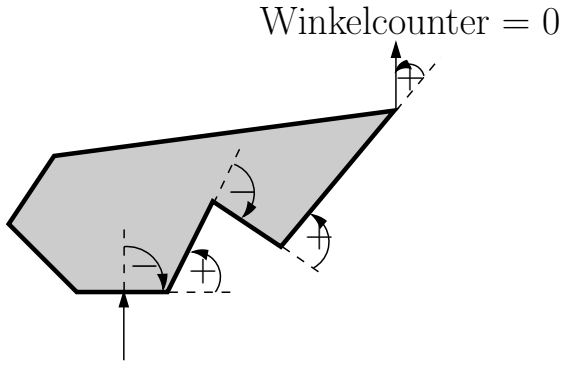


Entkommen aus dem Labyrinth: Modell

- Punktförmiger Agent/kreisförmiger Agent
- Touch Sensor
- Folge einer Wand
- Folge einer Richtung (exakt)
- Drehwinkel-Zähler
- Keinen weiteren Speicher

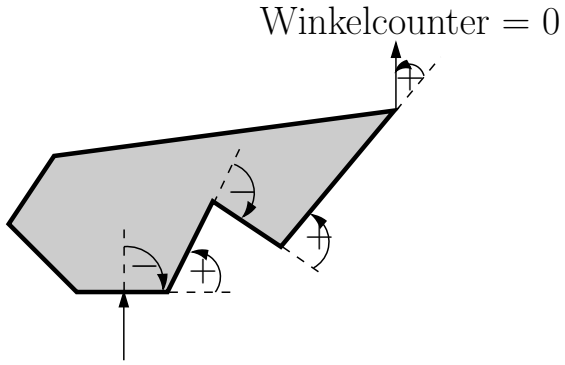


Pledge Algorithmus



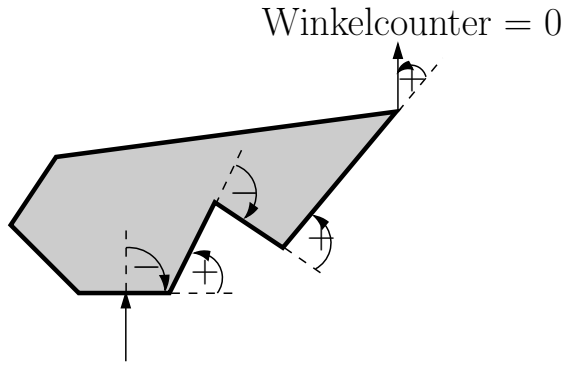
Pledge Algorithmus

1. Wähle Winkel φ und drehe den Roboter in diese Richtung.



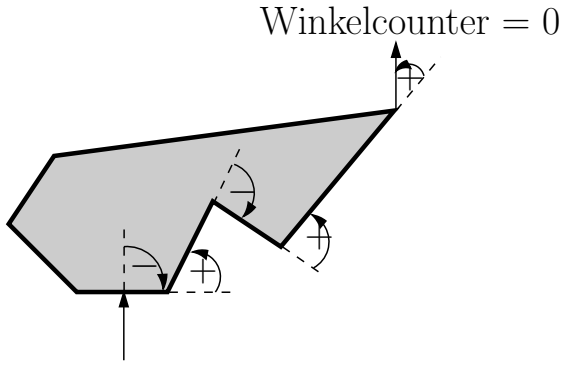
Pledge Algorithmus

1. Wähle Winkel φ und drehe den Roboter in diese Richtung.
2. Gehe in Richtung φ, bis der Roboter ein Hindernis erreicht.



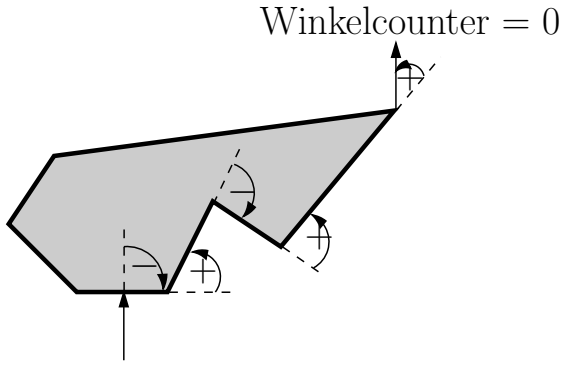
Pledge Algorithmus

1. Wähle Winkel φ und drehe den Roboter in diese Richtung.
2. Gehe in Richtung φ, bis der Roboter ein Hindernis erreicht.
3. Drehe nach rechts und halte den Kontakt mit der Wand an der linken Seite des Roboters.



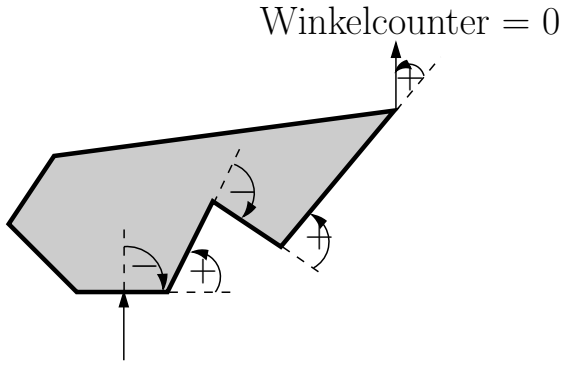
Pledge Algorithmus

1. Wähle Winkel φ und drehe den Roboter in diese Richtung.
2. Gehe in Richtung φ, bis der Roboter ein Hindernis erreicht.
3. Drehe nach rechts und halte den Kontakt mit der Wand an der linken Seite des Roboters.
4. Folge der Wand und addiere dabei die Drehwinkel, bis der totale Drehwinkel Null ist, dann GOTO (2).

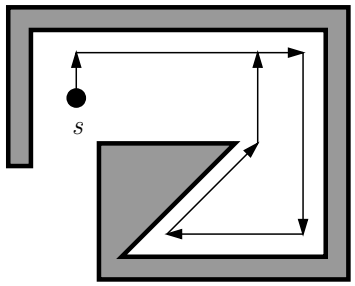


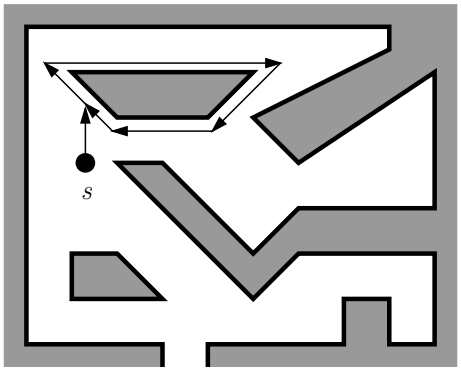
Pledge Algorithmus

1. Wähle Winkel φ und drehe den Roboter in diese Richtung.
2. Gehe in Richtung φ, bis der Roboter ein Hindernis erreicht.
3. Drehe nach rechts und halte den Kontakt mit der Wand an der linken Seite des Roboters.
4. Folge der Wand und addiere dabei die Drehwinkel, bis der totale Drehwinkel Null ist, dann GOTO (2).



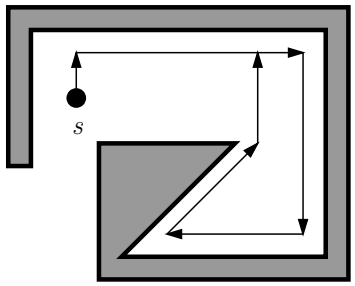
Pledge Algorithmus

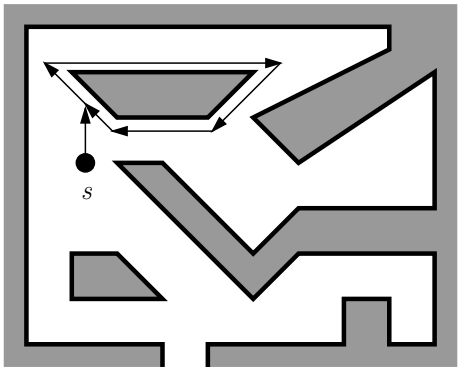




Pledge Algorithmus

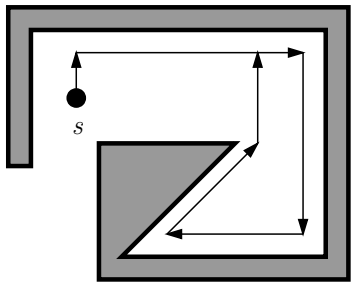
- Winkelzähler $\mod 2\pi = 0$ reicht nicht

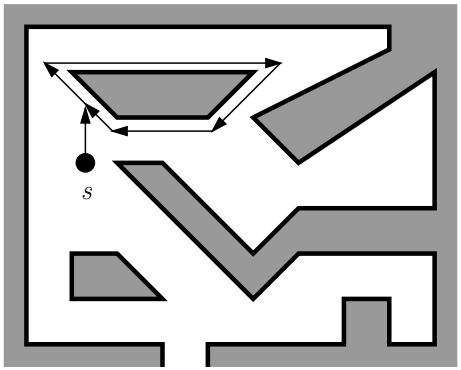




Pledge Algorithmus

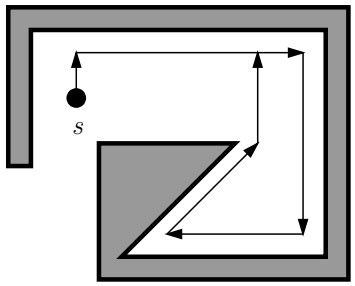
- Winkelzähler $\mathrm{mod}\ 2\pi = 0$ reicht nicht
- Nur Linke Hand reicht nicht

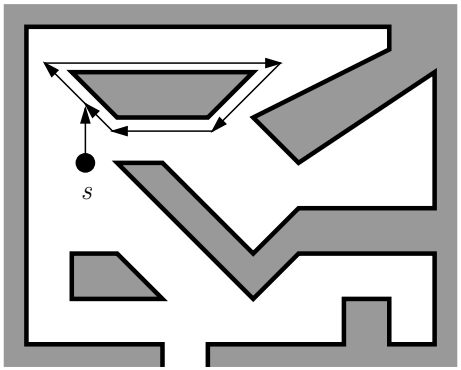




Pledge Algorithmus

- Winkelzähler $\mathrm{mod}\ 2\pi = 0$ reicht nicht
- Nur Linke Hand reicht nicht





Beweis Korrektheit: Winkelzähler nicht positiv

Beweis Korrektheit: Winkelzähler nicht positiv

Lemma 7.2 Der Winkelzähler im Pledge-Algorithmus nimmt niemals einen positiven Wert an.

Beweis Korrektheit: Winkelzähler nicht positiv

Lemma 7.2 Der Winkelzähler im Pledge-Algorithmus nimmt niemals einen positiven Wert an.

Beweis:

Beweis Korrektheit: Winkelzähler nicht positiv

Lemma 7.2 Der Winkelzähler im Pledge-Algorithmus nimmt niemals einen positiven Wert an.

Beweis:

- Zu Beginn Null

Beweis Korrektheit: Winkelzähler nicht positiv

Lemma 7.2 Der Winkelzähler im Pledge-Algorithmus nimmt niemals einen positiven Wert an.

Beweis:

- Zu Beginn Null
- Null bei Verlassen des Hindernisses

Beweis Korrektheit: Winkelzähler nicht positiv

Lemma 7.2 Der Winkelzähler im Pledge-Algorithmus nimmt niemals einen positiven Wert an.

Beweis:

- Zu Beginn Null
- Null bei Verlassen des Hindernisses
- Beim Auftreffen Rechtsdrehung $\Rightarrow$ negativ

Beweis Korrektheit: Winkelzähler nicht positiv

Lemma 7.2 Der Winkelzähler im Pledge-Algorithmus nimmt niemals einen positiven Wert an.

Beweis:

- Zu Beginn Null
- Null bei Verlassen des Hindernisses
- Beim Auftreffen Rechtsdrehung $\Rightarrow$ negativ
- Stetige Änderung: Null $\Rightarrow$ Weg frei

Beweis Korrektheit: Endlosstück

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

- Weg des Roboters ist polygonale Kette

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

- Weg des Roboters ist polygonale Kette
- Eckpunkte der Szene sind Knoten der Kette

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

- Weg des Roboters ist polygonale Kette
- Eckpunkte der Szene sind Knoten der Kette
- Auftreffpunkte auf Kanten sind Knoten der Kette

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

- Weg des Roboters ist polygonale Kette
- Eckpunkte der Szene sind Knoten der Kette
- Auftreffpunkte auf Kanten sind Knoten der Kette
- Zu jeder Ecke existiert genau ein Auftreffpunkt auf Kante

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

- Weg des Roboters ist polygonale Kette
- Eckpunkte der Szene sind Knoten der Kette
- Auftreffpunkte auf Kanten sind Knoten der Kette
- Zu jeder Ecke existiert genau ein Auftreffpunkt auf Kante
- Endliche Menge S von möglichen Ecken für Weg

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

- Weg des Roboters ist polygonale Kette
- Eckpunkte der Szene sind Knoten der Kette
- Auftreffpunkte auf Kanten sind Knoten der Kette
- Zu jeder Ecke existiert genau ein Auftreffpunkt auf Kante
- Endliche Menge S von möglichen Ecken für Weg
- Gleicher Zählerstand an Ecke $\Rightarrow$ gleicher Weg immer wieder

Beweis Korrektheit: Endlosstück

Lemma 7.3 Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

Beweis:

- Weg des Roboters ist polygonale Kette
- Eckpunkte der Szene sind Knoten der Kette
- Auftreffpunkte auf Kanten sind Knoten der Kette
- Zu jeder Ecke existiert genau ein Auftreffpunkt auf Kante
- Endliche Menge S von möglichen Ecken für Weg
- Gleicher Zählerstand an Ecke $\Rightarrow$ gleicher Weg immer wieder
- Annahme: Nie gleicher Zählerstand

- 1. Fall: Löst sich irgendwann nicht mehr $\Rightarrow$ gleicher Weg immer wieder

- 1. Fall: Löst sich irgendwann nicht mehr $\Rightarrow$ gleicher Weg immer wieder
- 2. Fall: Löst sich mehr als $|S|$ mal (unendlich oft)

- 1. Fall: Löst sich irgendwann nicht mehr $\Rightarrow$ gleicher Weg immer wieder
- 2. Fall: Löst sich mehr als $|S|$ mal (unendlich oft)
- $\Rightarrow$ zweimal mit gleichem Zählerstand 0 an gleicher Ecke, Widerspruch!

Beweis Korrektheit: Endlosweg schnittfrei

Beweis Korrektheit: Endlosweg schnittfrei

Lemma 7.4 Angenommen, der Roboter kann dem Labyrinth nicht entkommen. Sei $\Pi_\circ$ der Teil des Weges, den der Roboter immer wieder durchläuft. $\Pi_\circ$ ist kreuzungsfrei.

Beweis Korrektheit: Endlosweg schnittfrei

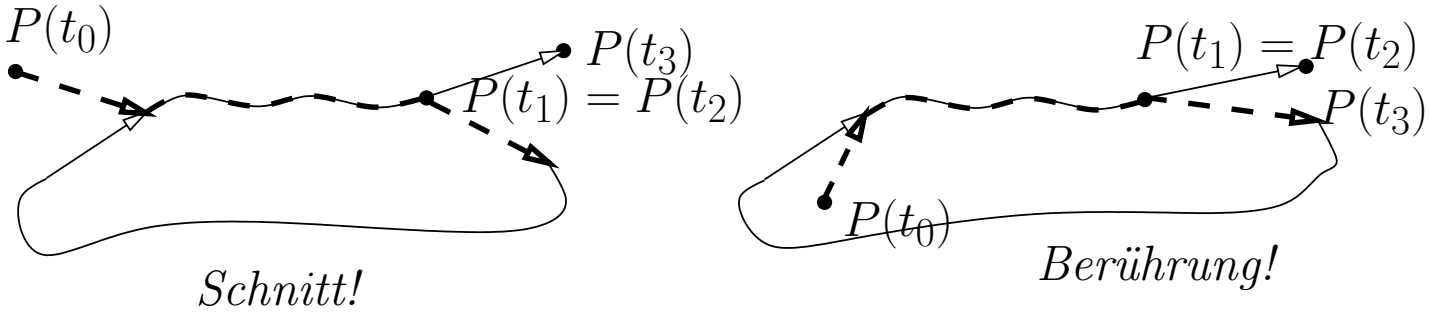
Lemma 7.4 Angenommen, der Roboter kann dem Labyrinth nicht entkommen. Sei Π_∞ der Teil des Weges, den der Roboter immer wieder durchläuft. Π_∞ ist kreuzungsfrei.

Unterschied: Kreuzen/Berühren

Beweis Korrektheit: Endlosweg schnittfrei

Lemma 7.4 Angenommen, der Roboter kann dem Labyrinth nicht entkommen. Sei $\Pi_\circ$ der Teil des Weges, den der Roboter immer wieder durchläuft. $\Pi_\circ$ ist kreuzungsfrei.

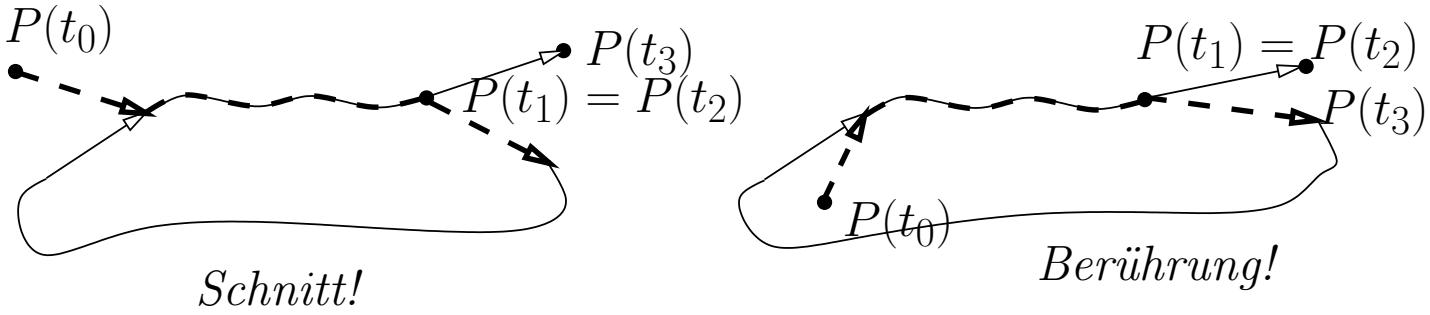
Unterschied: Kreuzen/Berühren



Beweis Korrektheit: Endlosweg schnittfrei

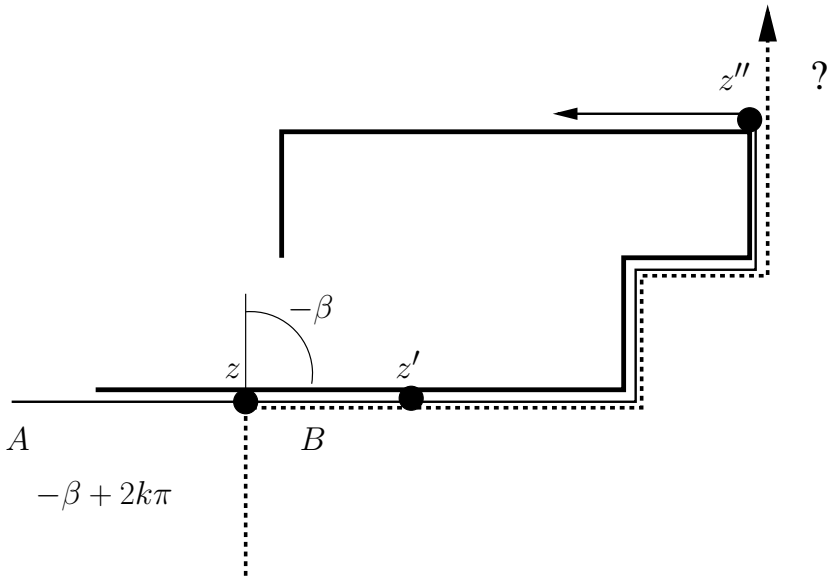
Lemma 7.4 Angenommen, der Roboter kann dem Labyrinth nicht entkommen. Sei $\Pi_\circ$ der Teil des Weges, den der Roboter immer wieder durchläuft. $\Pi_\circ$ ist kreuzungsfrei.

Unterschied: Kreuzen/Berühren



Kreuzung nur am Hindernis, freie Wege parallel!

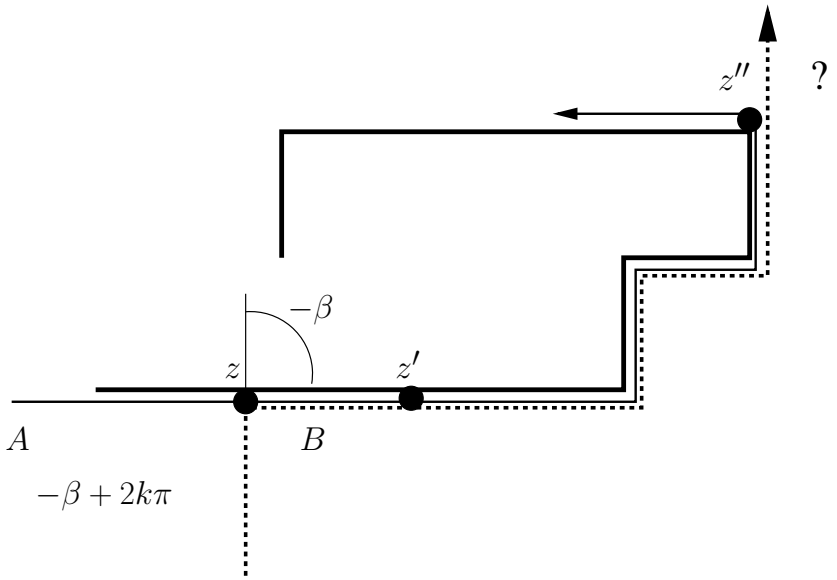
Beweis Korrektheit: Endlosweg schnittfrei



Beweis Korrektheit: Endlosweg schnittfrei

Lemma 7.4 Annahme Roboter kann nicht entkommen. $\Pi_\circ$ stets aufs Neue. $\Pi_\circ$ ist kreuzungsfrei.

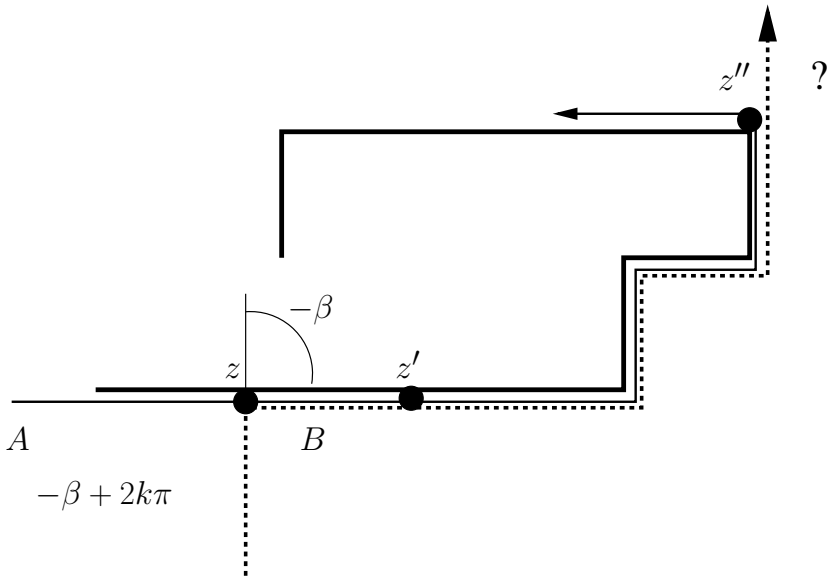
- Beweis:



Beweis Korrektheit: Endlosweg schnittfrei

Lemma 7.4 Annahme Roboter kann nicht entkommen. $\Pi_\circ$ stets aufs Neue. $\Pi_\circ$ ist kreuzungsfrei.

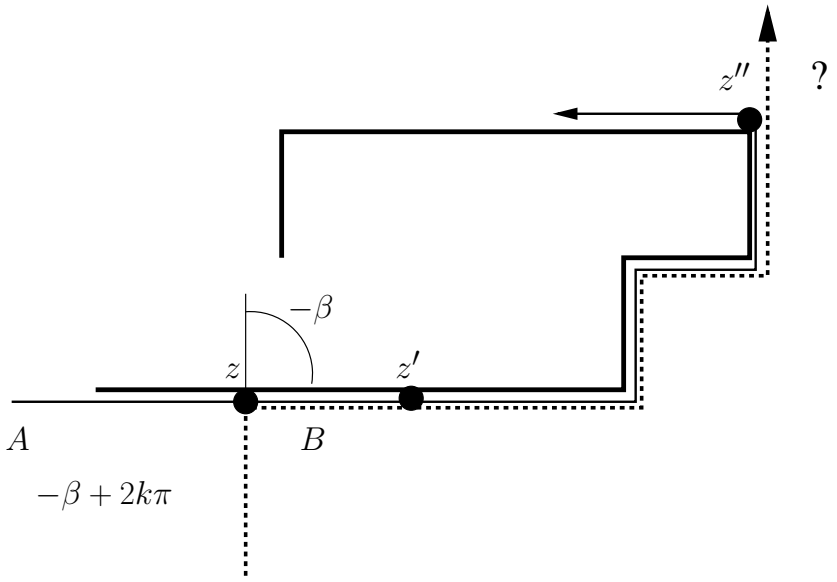
- Beweis: Einer der beiden Segmente z.B. B ist frei



Beweis Korrektheit: Endlosweg schnittfrei

Lemma 7.4 Annahme Roboter kann nicht entkommen. $\Pi_\circ$ stets aufs Neue. $\Pi_\circ$ ist kreuzungsfrei.

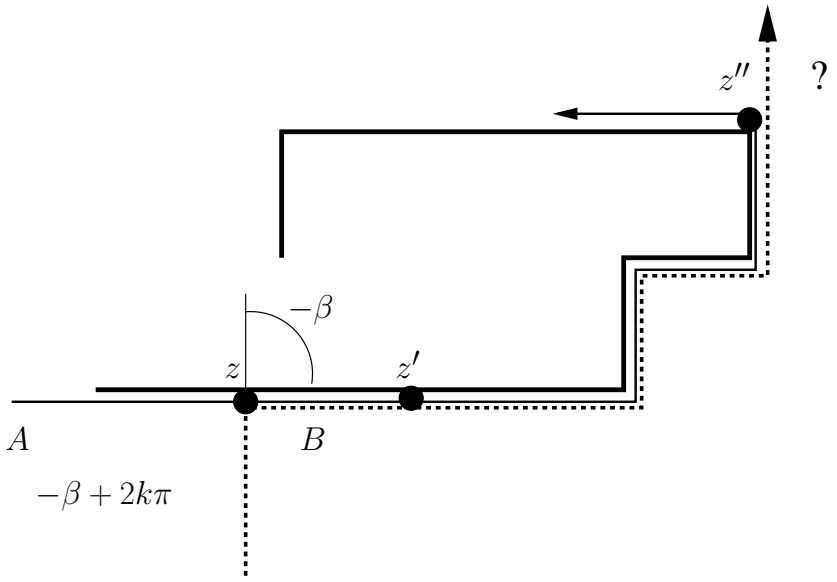
- Beweis: Einer der beiden Segmente z.B. B ist frei
- Kurz hinter z Winkelzähler $C_A(z')$ und $C_B(z')$



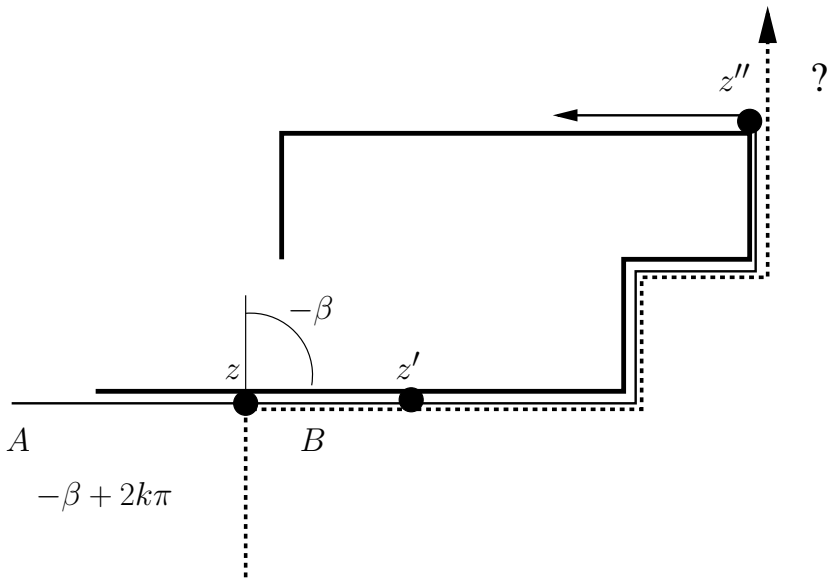
Beweis Korrektheit: Endlosweg schnittfrei

Lemma 7.4 Annahme Roboter kann nicht entkommen. $\Pi_\circ$ stets aufs Neue. $\Pi_\circ$ ist kreuzungsfrei.

- Beweis: Einer der beiden Segmente z.B. B ist frei
- Kurz hinter z Winkelzähler $C_A(z')$ und $C_B(z')$
- $C_B(z') = -\beta$ und $C_A(z') = -\beta + 2k\pi$ mit $k \in Z$

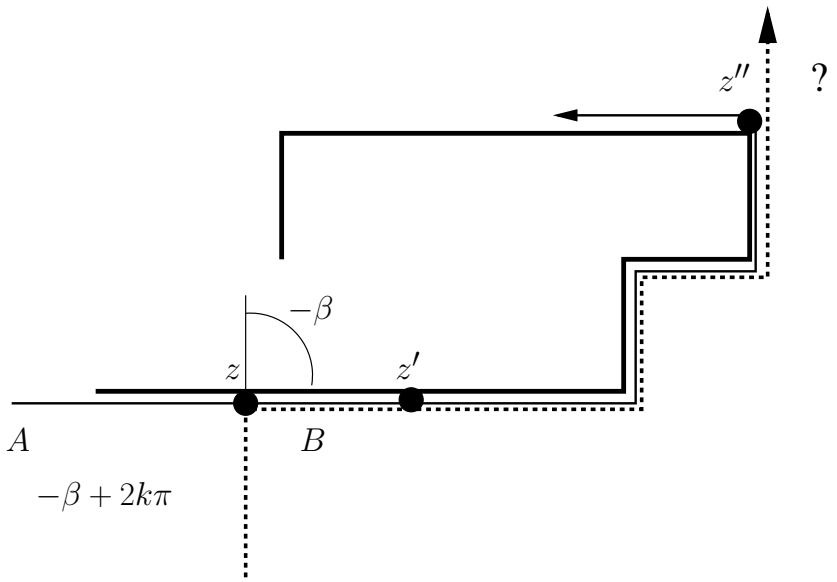


Beweis Lemma 7.4 Endlosweg schnittfrei



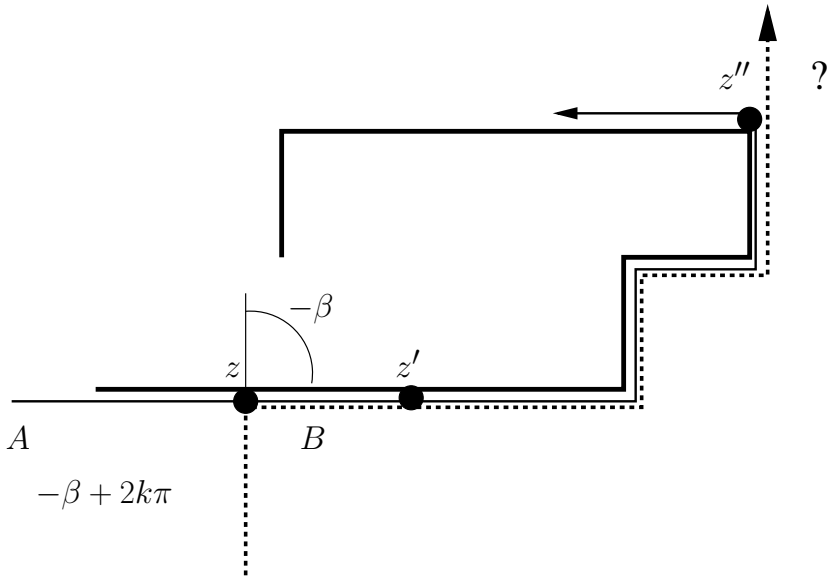
Beweis Lemma 7.4 Endlosweg schnittfrei

- $C_B(z') = -\beta$ und $C_A(z') = -\beta + 2k\pi$ mit $k \in Z$
- $k = 0$? Geht nicht wg. determ. Widerspruch!



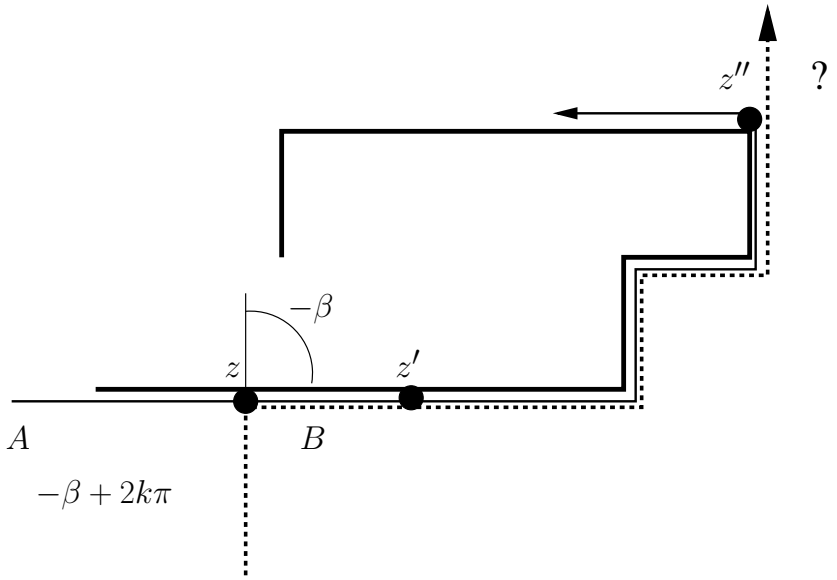
Beweis Lemma 7.4 Endlosweg schnittfrei

- $C_B(z') = -\beta$ und $C_A(z') = -\beta + 2k\pi$ mit $k \in Z$
- $k = 0$? Geht nicht wg. determ. Widerspruch!
- $k > 0$? Geht nicht wg. Lemma 7.2, $C_A(z')$ negativ



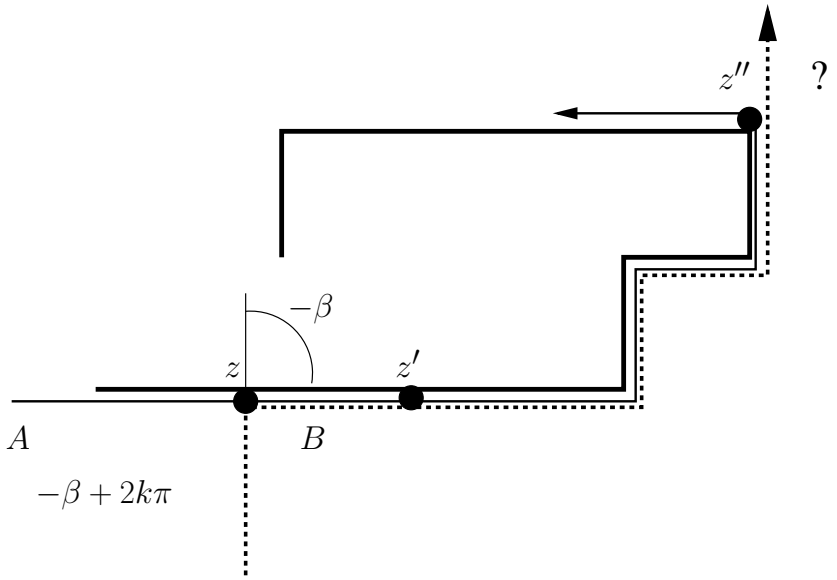
Beweis Lemma 7.4 Endlosweg schnittfrei

- $C_B(z') = -\beta$ und $C_A(z') = -\beta + 2k\pi$ mit $k \in Z$
- $k = 0$? Geht nicht wg. determ. Widerspruch!
- $k > 0$? Geht nicht wg. Lemma 7.2, $C_A(z')$ negativ
- Also $k < 0$ dann $C_A(p) < C_B(p)$ für alle p zwischen z' und z''



Beweis Lemma 7.4 Endlosweg schnittfrei

- $C_B(z') = -\beta$ und $C_A(z') = -\beta + 2k\pi$ mit $k \in Z$
- $k = 0$? Geht nicht wg. determ. Widerspruch!
- $k > 0$? Geht nicht wg. Lemma 7.2, $C_A(z')$ negativ
- Also $k < 0$ dann $C_A(p) < C_B(p)$ für alle p zwischen z' und z''
- Weg B trennt sich zuerst, kein Schnitt!!!



Beweis Korrektheit!

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue
- **Lemma 7.4:** Ohne echte Kreuzungen

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue
- **Lemma 7.4:** Ohne echte Kreuzungen
- Zwei Orientierungen: 1) Im UZS 2) Gegen den UZS

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue
- **Lemma 7.4:** Ohne echte Kreuzungen
- Zwei Orientierungen: 1) Im UZS 2) Gegen den UZS
- 2) stets $+2\pi$ pro Runde, irgendwann positiv, Widerspruch

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue
- **Lemma 7.4:** Ohne echte Kreuzungen
- Zwei Orientierungen: 1) Im UZS 2) Gegen den UZS
- 2) stets $+2\pi$ pro Runde, irgendwann positiv, Widerspruch
- Also 1) stets -2π pro Runde

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue
- **Lemma 7.4:** Ohne echte Kreuzungen
- Zwei Orientierungen: 1) Im UZS 2) Gegen den UZS
- 2) stets $+2\pi$ pro Runde, irgendwann positiv, Widerspruch
- Also 1) stets -2π pro Runde
- Irgendwann nur noch negativ, Hindernis wird nicht verlassen

Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

Beweis:

- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue
- **Lemma 7.4:** Ohne echte Kreuzungen
- Zwei Orientierungen: 1) Im UZS 2) Gegen den UZS
- 2) stets $+2\pi$ pro Runde, irgendwann positiv, Widerspruch
- Also 1) stets -2π pro Runde
- Irgendwann nur noch negativ, Hindernis wird nicht verlassen
- Orientierung: Im UZS $\Rightarrow$ Innenhof!