

Offline Bewegungsplanung: TPP und Wege in 3D

Elmar Langetepe
University of Bonn

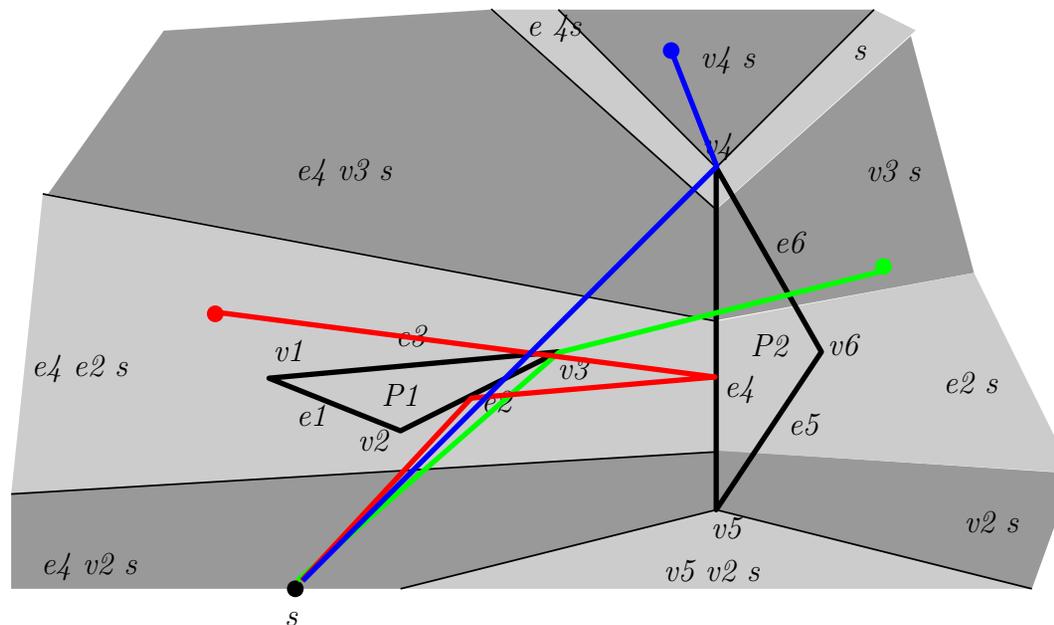
Last step shortest path map

Last step shortest path map

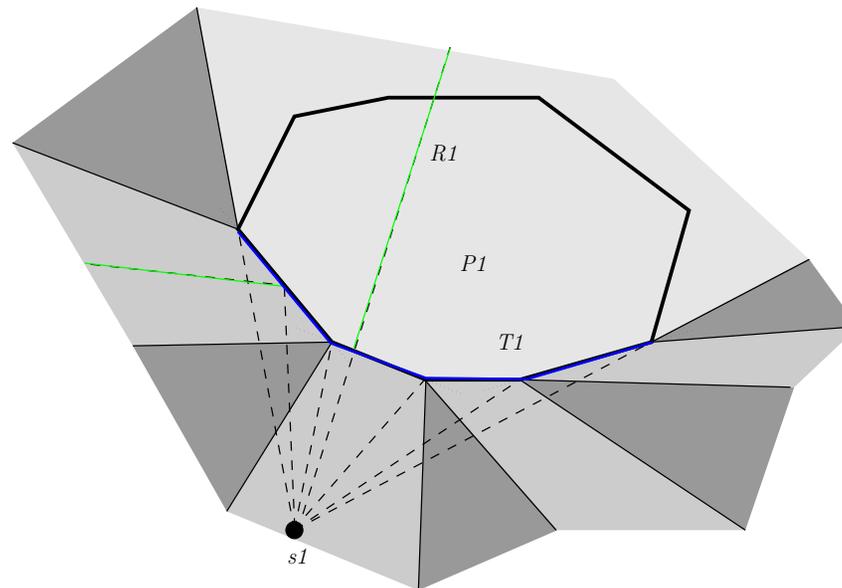
- Der letzte Schritt des Kürzesten Weges

Last step shortest path map

- Der letzte Schritt des Kürzesten Weges

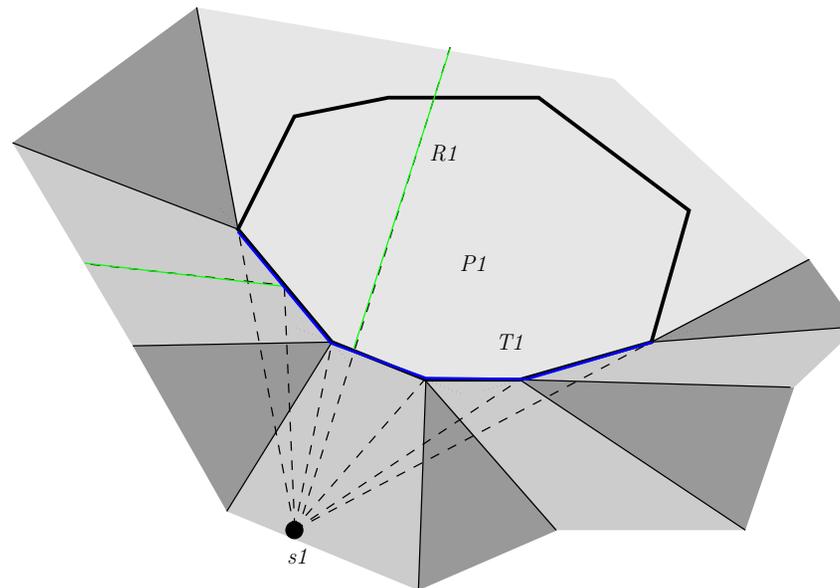


Last Step SPM S_1, S_2, \dots, S_k



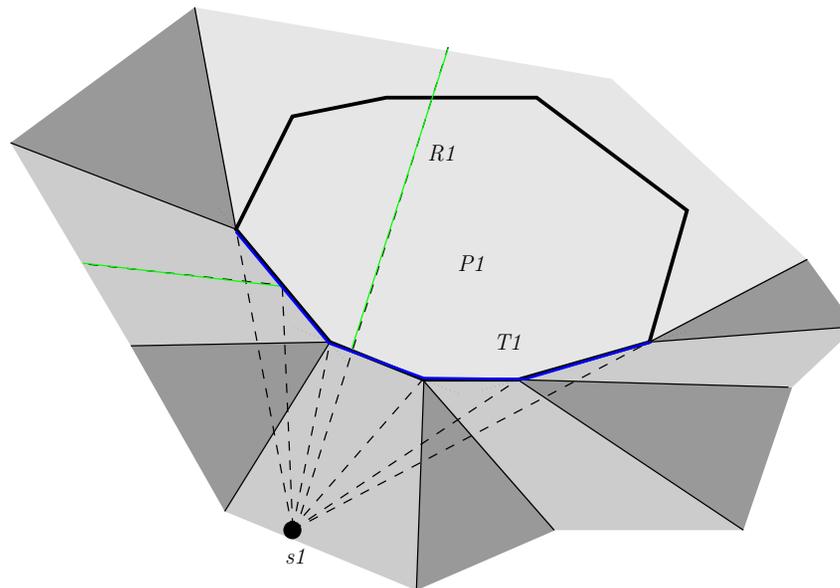
Last Step SPM S_1, S_2, \dots, S_k

- S_i gehört zur Sequenz P_1, P_2, \dots, P_i



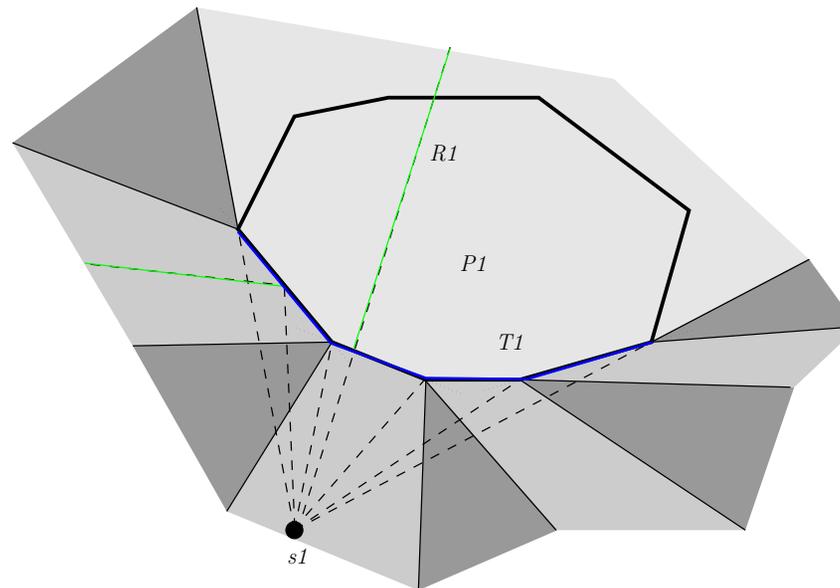
Last Step SPM S_1, S_2, \dots, S_k

- S_i gehört zur Sequenz P_1, P_2, \dots, P_i
- Beschaffenheit S_i : Reflektionsbereich T_i (konvexe Kette)



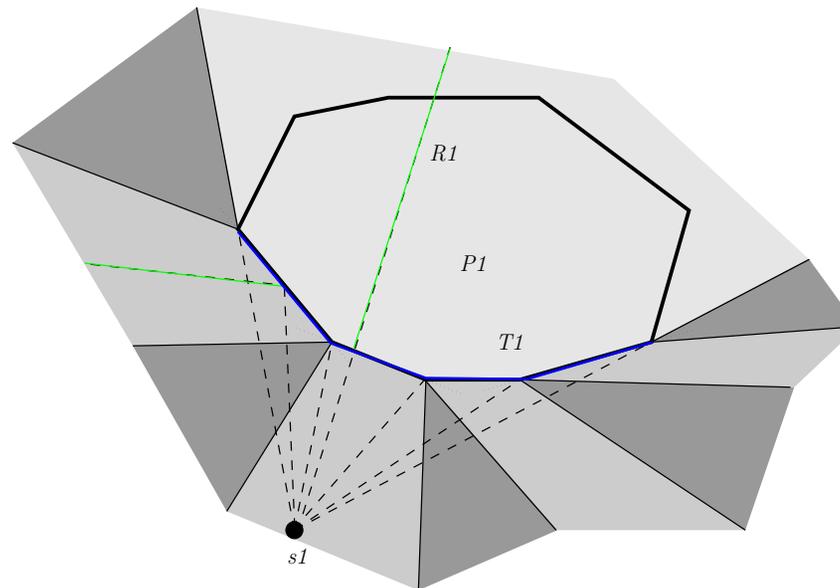
Last Step SPM S_1, S_2, \dots, S_k

- S_i gehört zur Sequenz P_1, P_2, \dots, P_i
- Beschaffenheit S_i : Reflektionsbereich T_i (konvexe Kette)
- Ausgehende Strahlen: Stern R_i , disjunkt



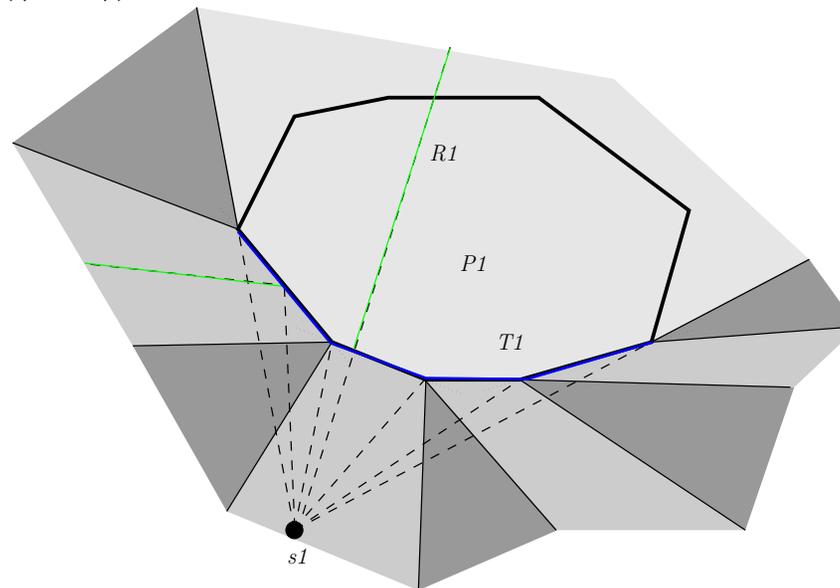
Last Step SPM S_1, S_2, \dots, S_k

- S_i gehört zur Sequenz P_1, P_2, \dots, P_i
- Beschaffenheit S_i : Reflektionsbereich T_i (konvexe Kette)
- Ausgehende Strahlen: Stern R_i , disjunkt
- Jeder Punkt wird von genau einem Strahl getroffen



Last Step SPM S_1, S_2, \dots, S_k

- S_i gehört zur Sequenz P_1, P_2, \dots, P_i
- Beschaffenheit S_i : Reflektionsbereich T_i (konvexe Kette)
- Ausgehende Strahlen: Stern R_i , disjunkt
- Jeder Punkt wird von genau einem Strahl getroffen
- Komplexität: $O(|P_i|)$



Lemma 1.32: Beweis!

Lemma 1.32: Beweis!

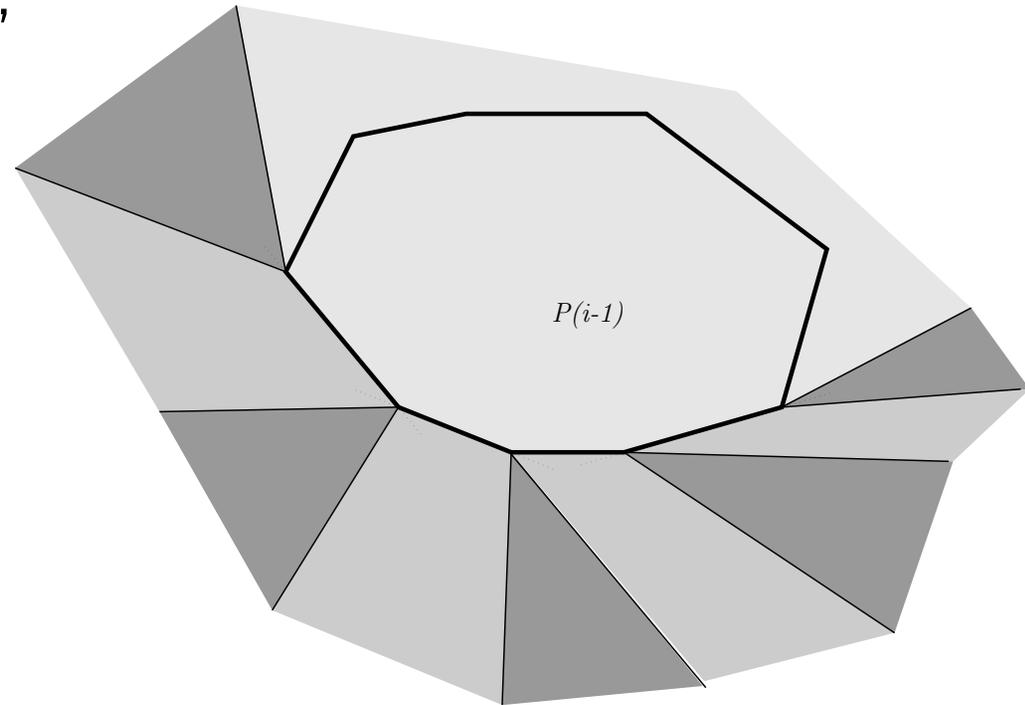
- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i

Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1

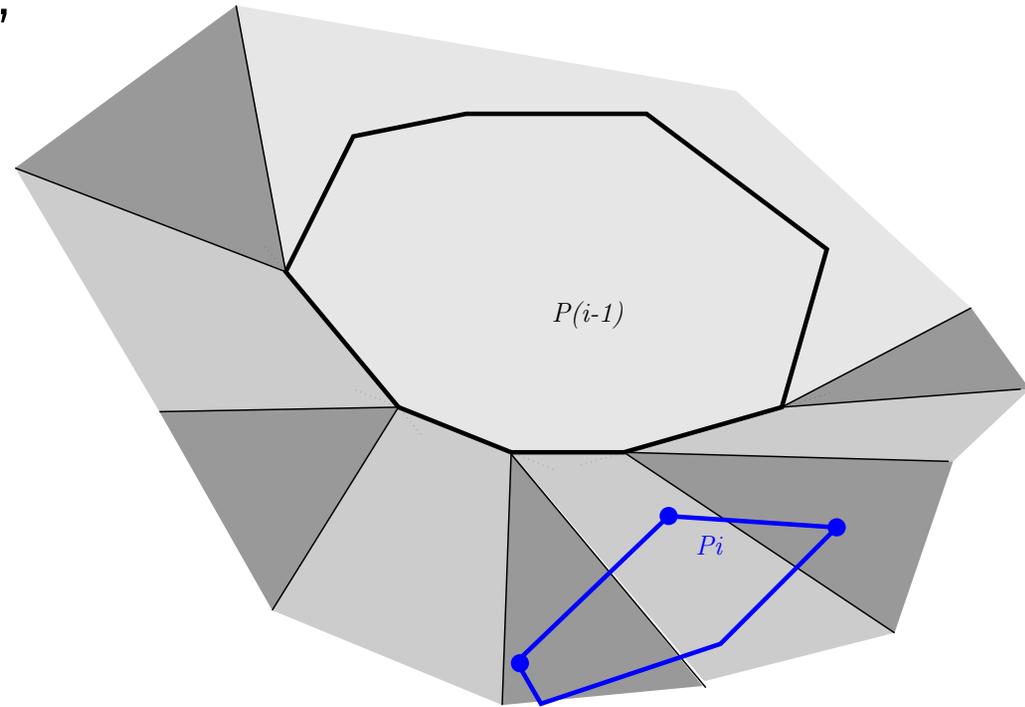
Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1
- Annahme: Gilt für
 S_1, S_2, \dots, S_{i-1}



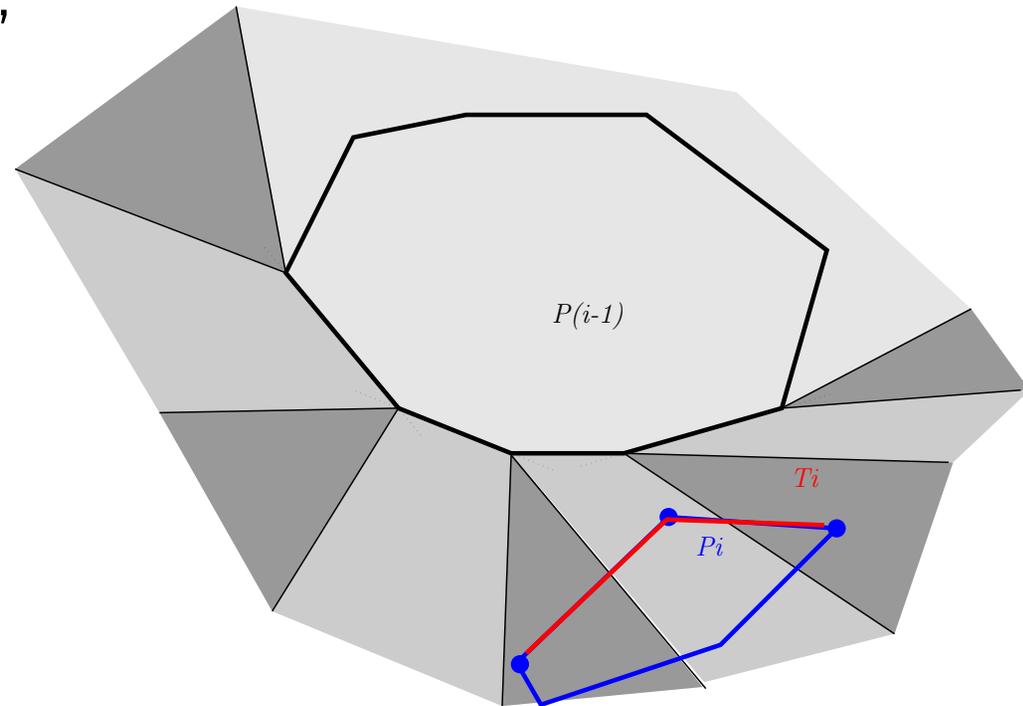
Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1
- Annahme: Gilt für
 S_1, S_2, \dots, S_{i-1}
- Lege konvexes Polygon P_i
in S_{i-1}



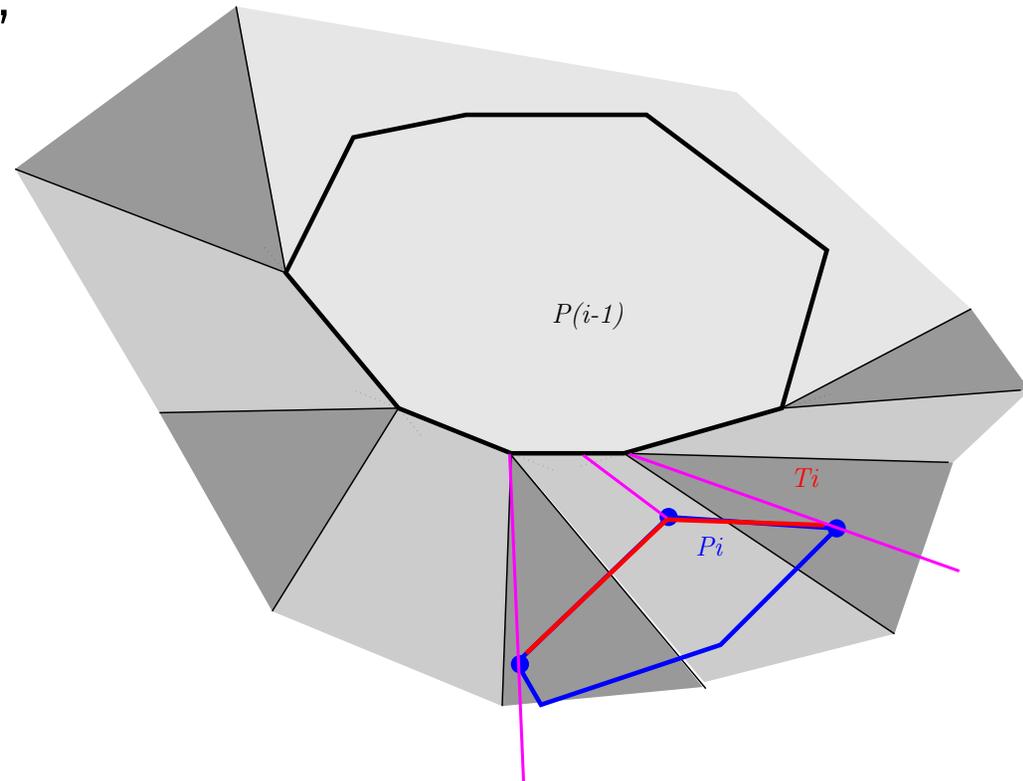
Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1
- Annahme: Gilt für
 S_1, S_2, \dots, S_{i-1}
- Lege konvexes Polygon P_i
in S_{i-1}
- *Sichtbare* Eckpunkte
konvexer Kette von
disjunkten Strahlen getroffen



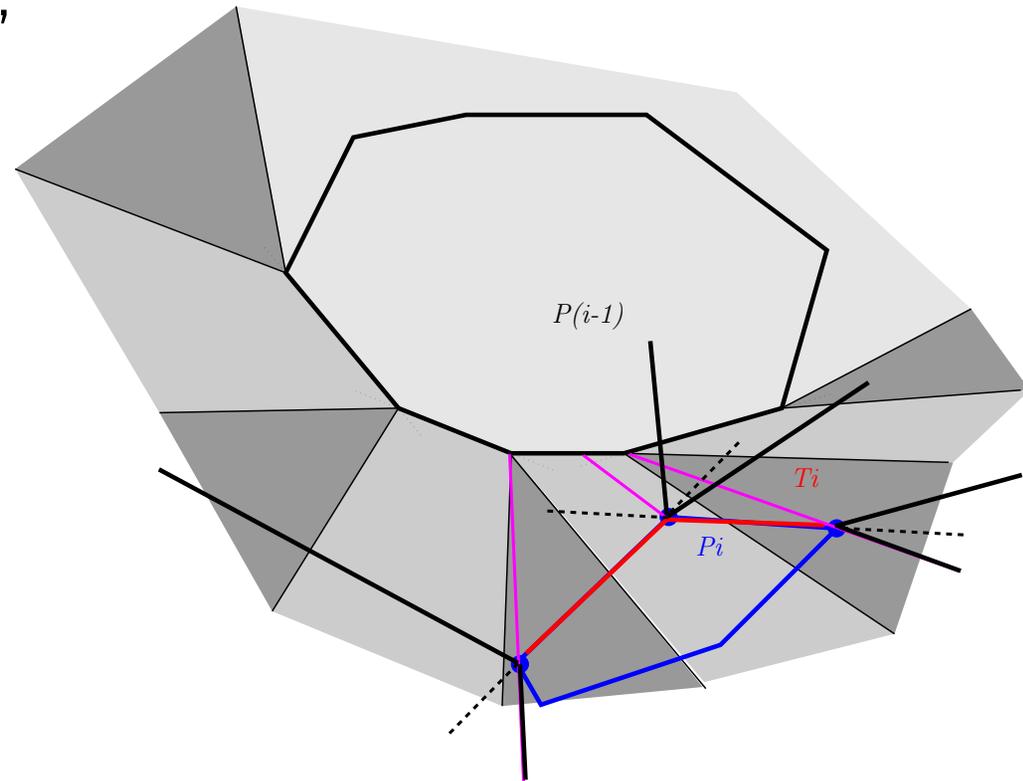
Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1
- Annahme: Gilt für
 S_1, S_2, \dots, S_{i-1}
- Lege konvexes Polygon P_i
in S_{i-1}
- *Sichtbare* Eckpunkte
konvexer Kette von
disjunkten Strahlen getroffen



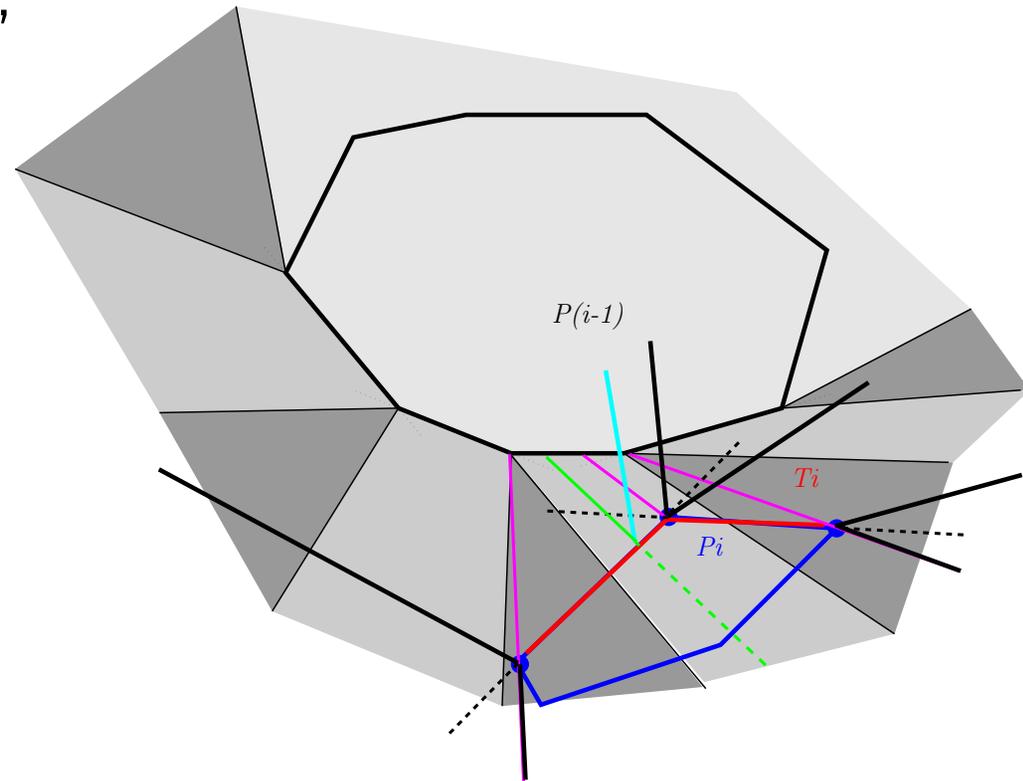
Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1
- Annahme: Gilt für
 S_1, S_2, \dots, S_{i-1}
- Lege konvexes Polygon P_i
in S_{i-1}
- *Sichtbare* Eckpunkte
konvexer Kette von
disjunkten Strahlen getroffen
- Reflektionen disjunkt



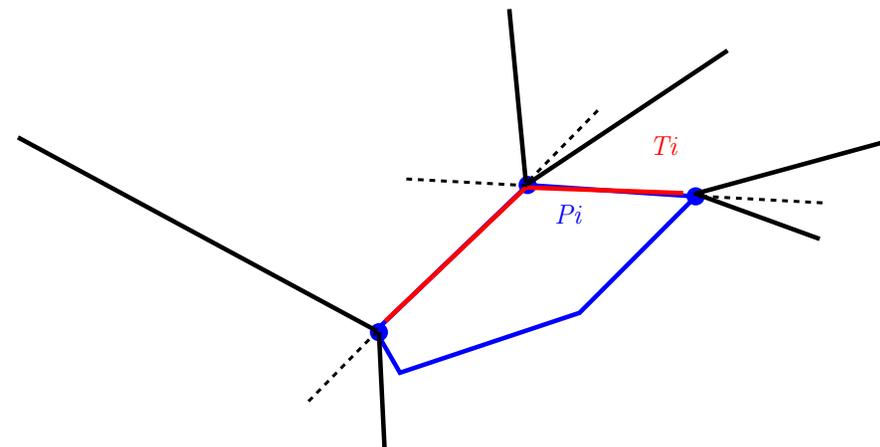
Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1
- Annahme: Gilt für
 S_1, S_2, \dots, S_{i-1}
- Lege konvexes Polygon P_i
in S_{i-1}
- *Sichtbare* Eckpunkte
konvexer Kette von
disjunkten Strahlen getroffen
- Reflektionen disjunkt



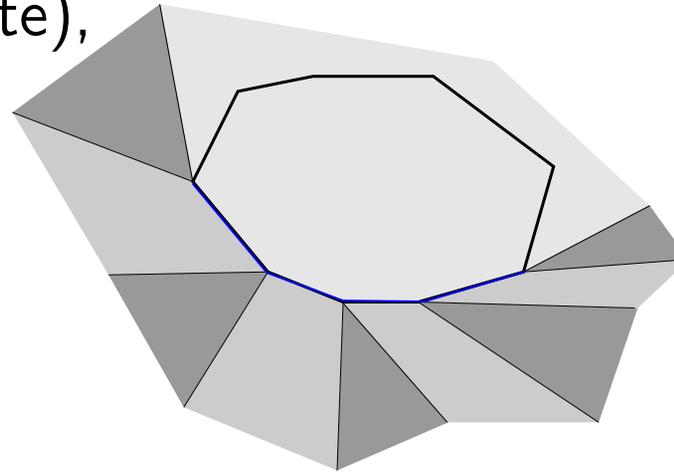
Lemma 1.32: Beweis!

- Ind.: $O(|P_i|)$, Konvexe Kette T_i ,
Disjunkter Stern R_i
- Gilt für S_1
- Annahme: Gilt für
 S_1, S_2, \dots, S_{i-1}
- Lege konvexes Polygon P_i
in S_{i-1}
- *Sichtbare* Eckpunkte
konvexer Kette von
disjunkten Strahlen getroffen
- Reflektionen disjunkt



DS und Komplexität für S_i !

- Vorbereitet für Lokalisation
- Reflektionsbereich(Knoten/Kante),
Durchgangsbereich
- Konvexe Kette,
 n_i disjunkte Strahlen
- Balancierter Baum:
Lokalization in $O(\log n_i)$
- Für alle S_i gleich



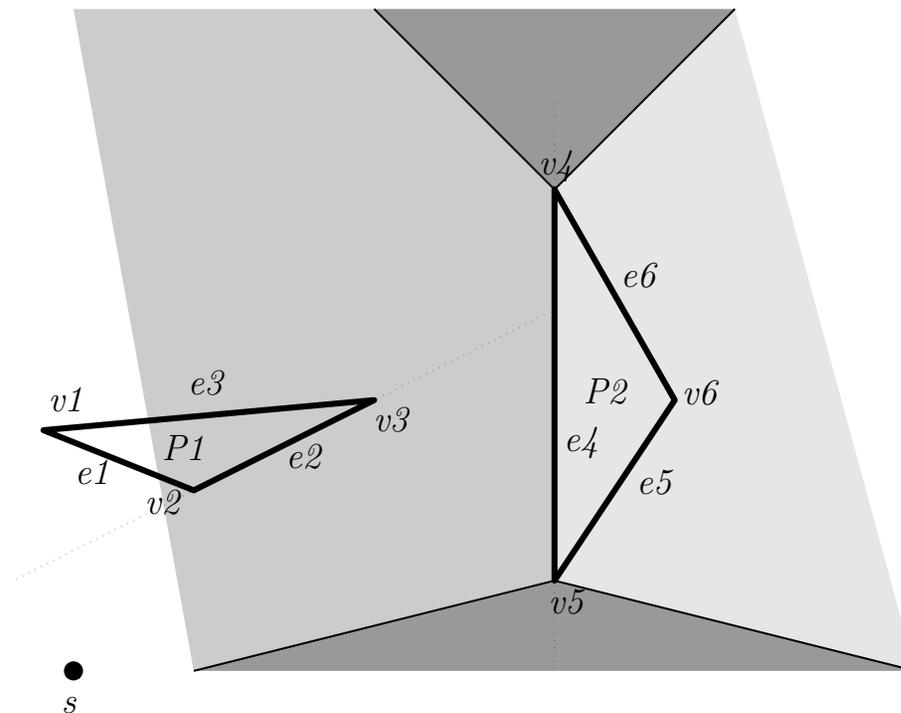
Benutzung der Last Step SPM: Alg. 1.10

Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k

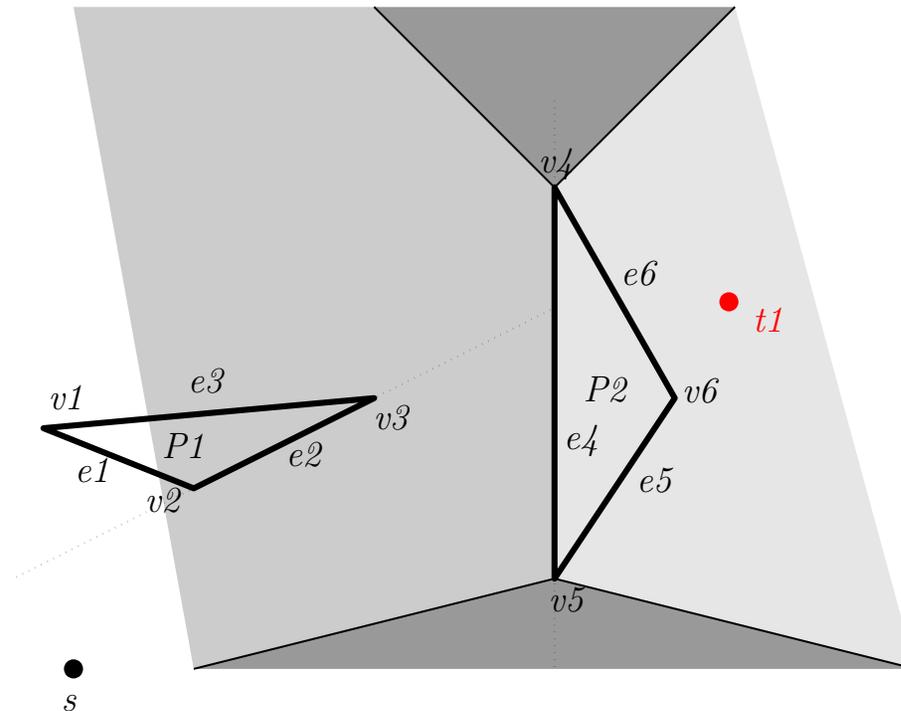
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



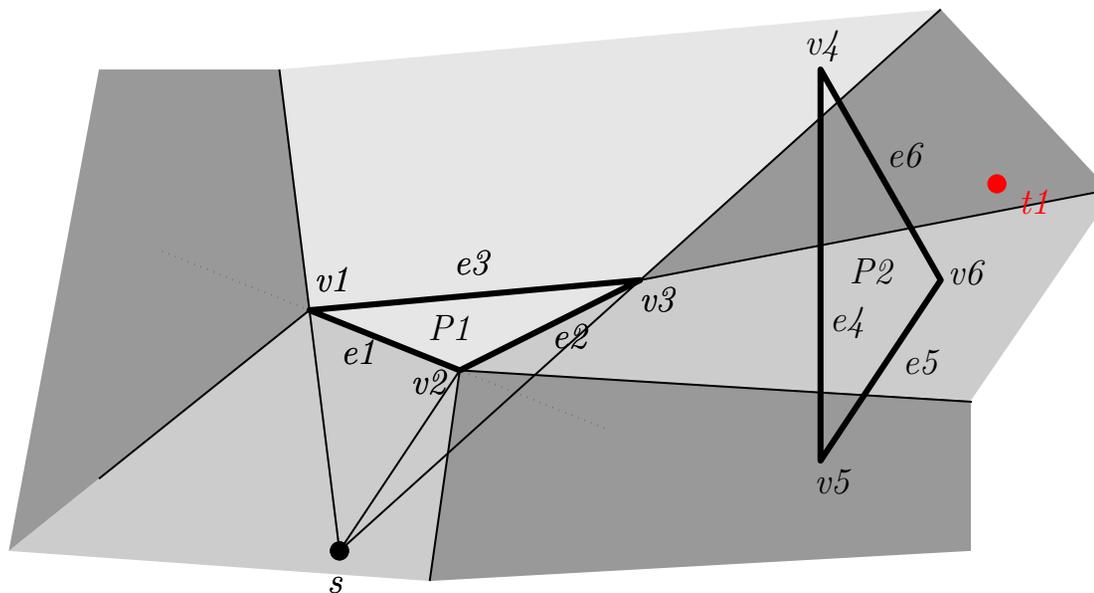
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



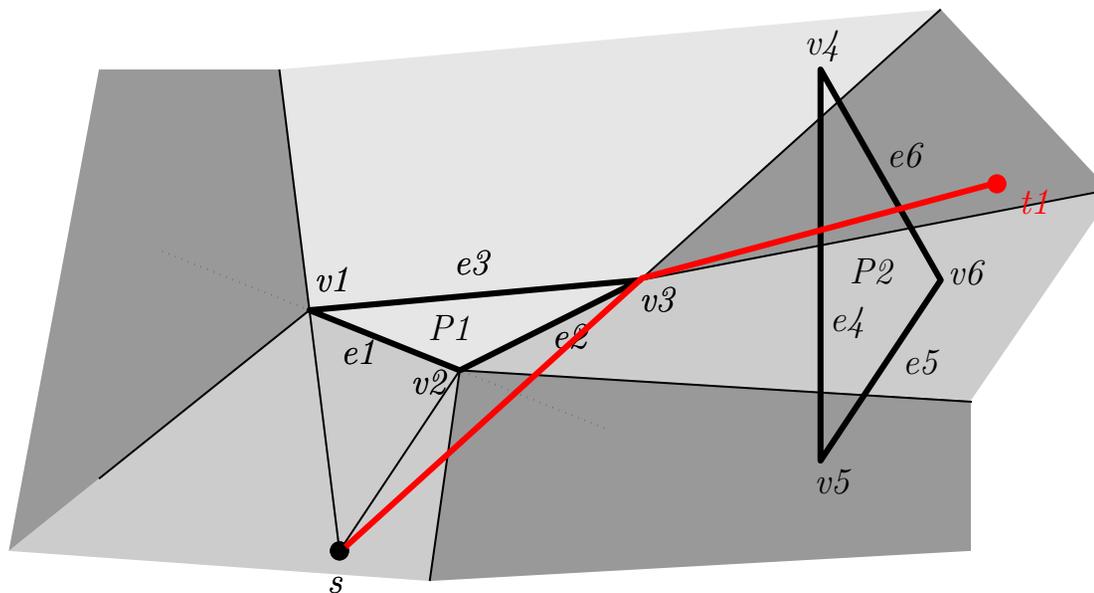
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



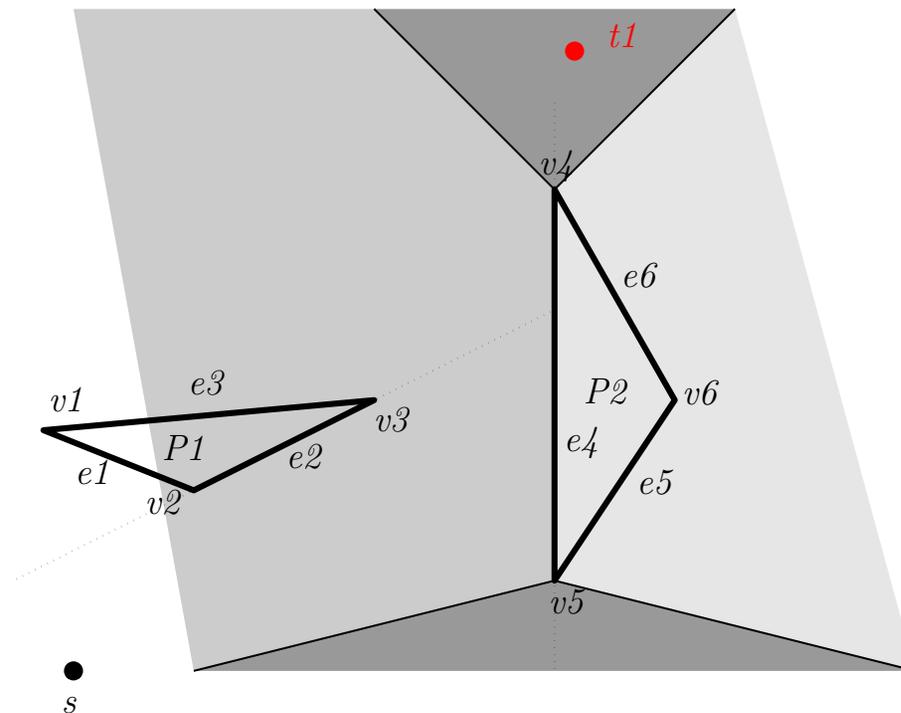
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



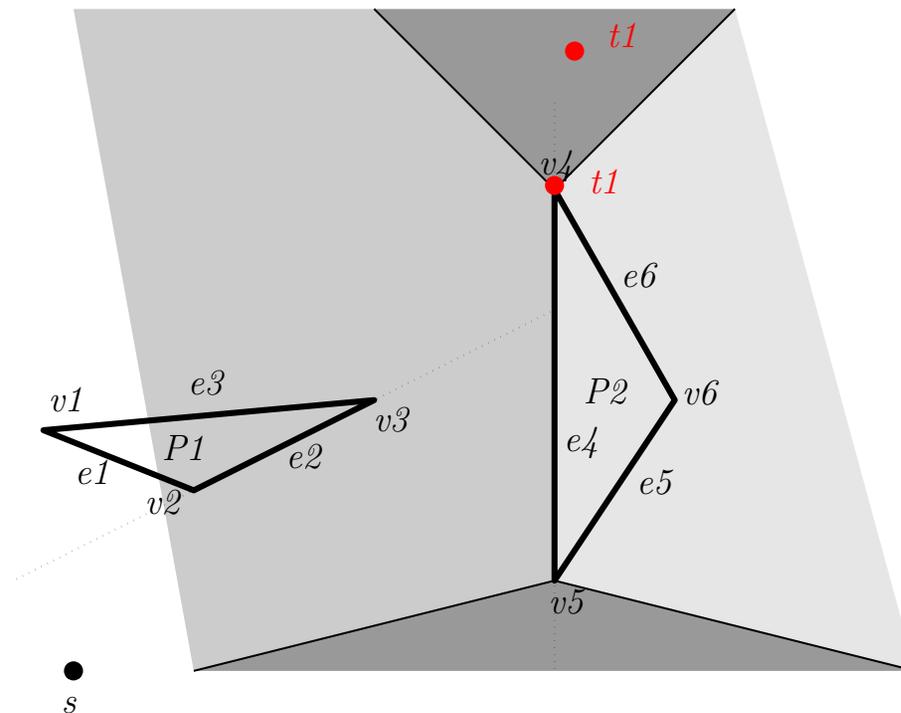
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



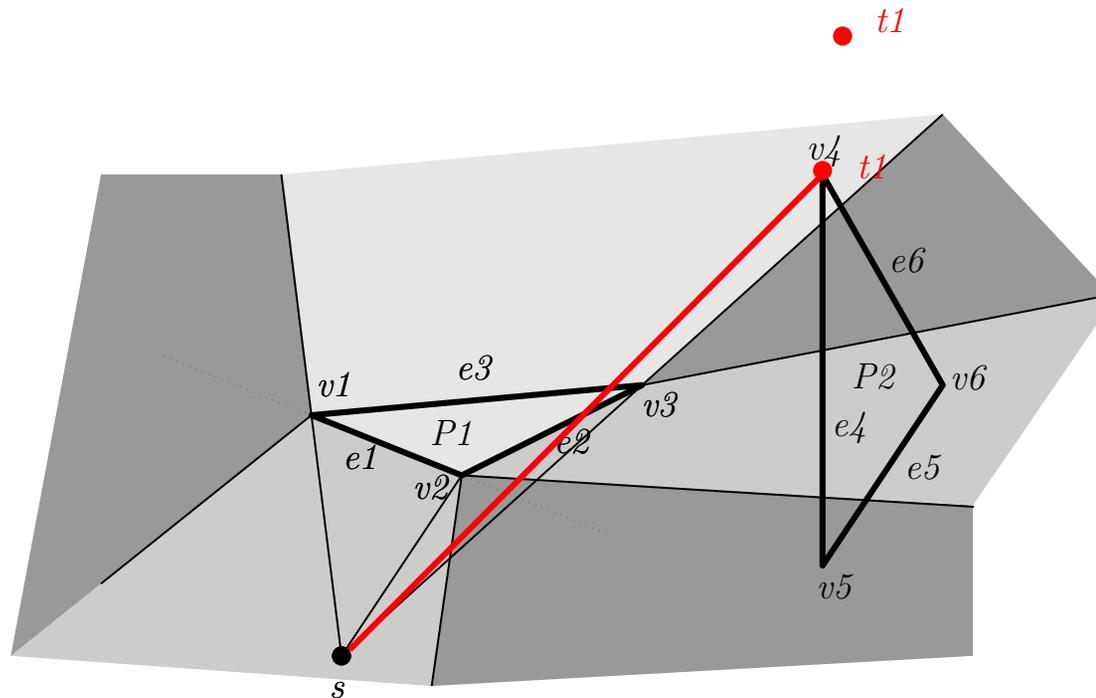
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



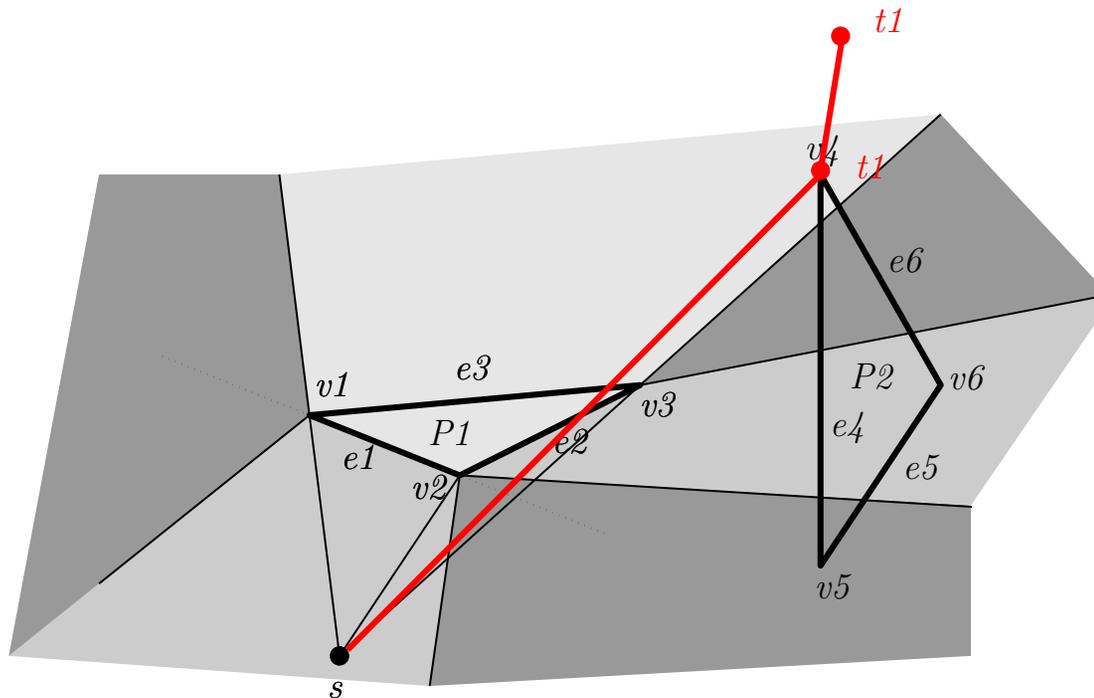
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



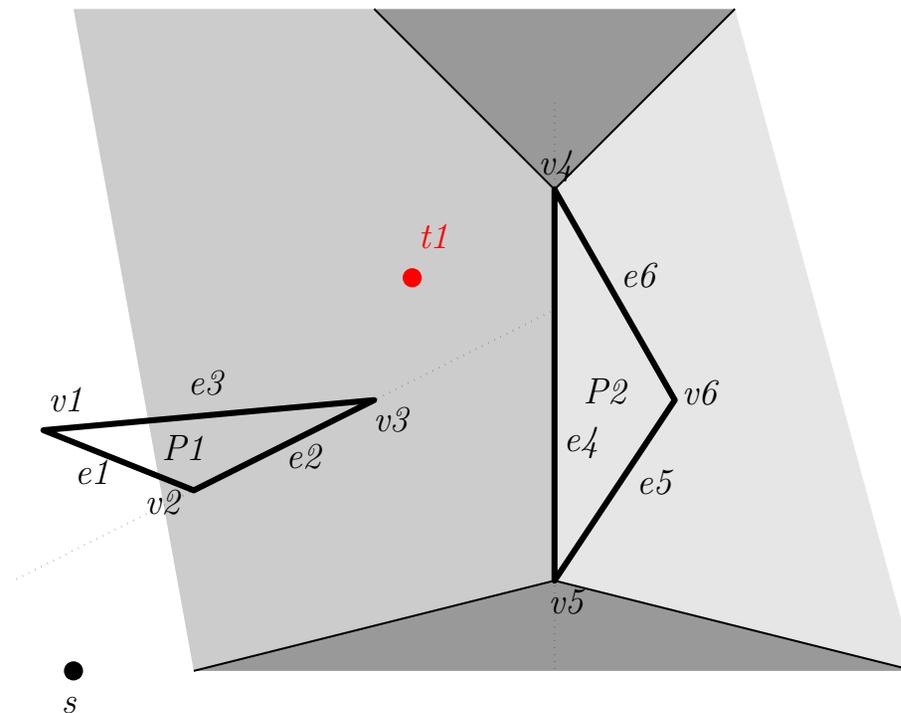
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



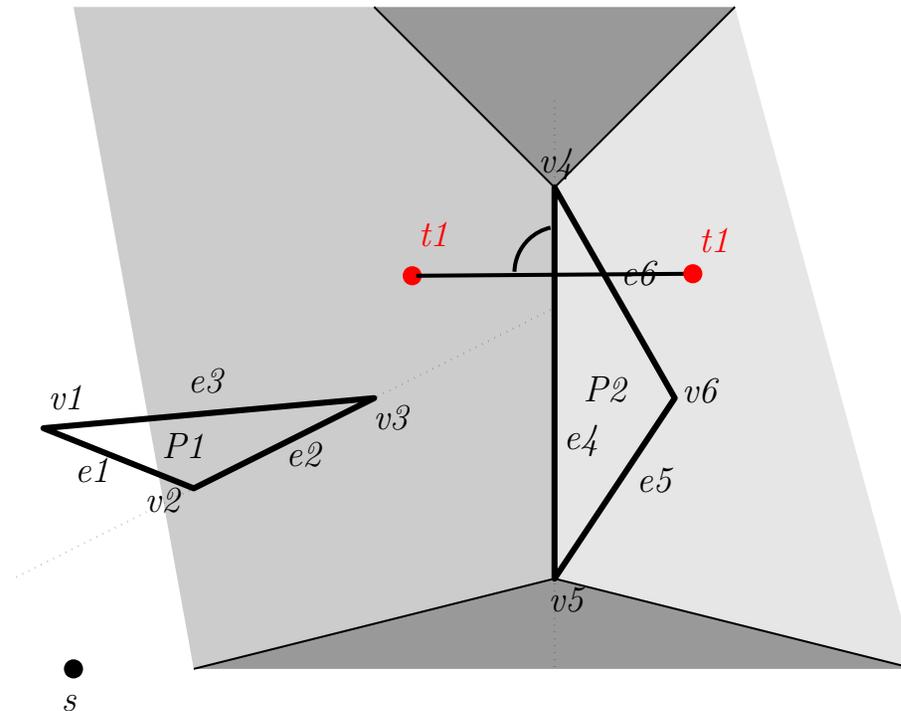
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



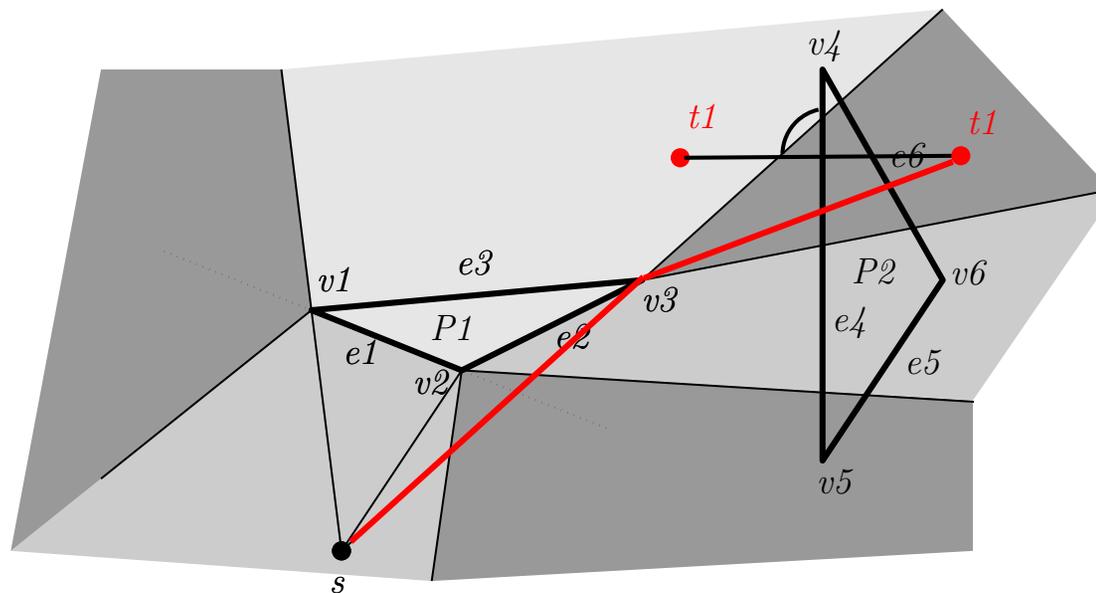
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



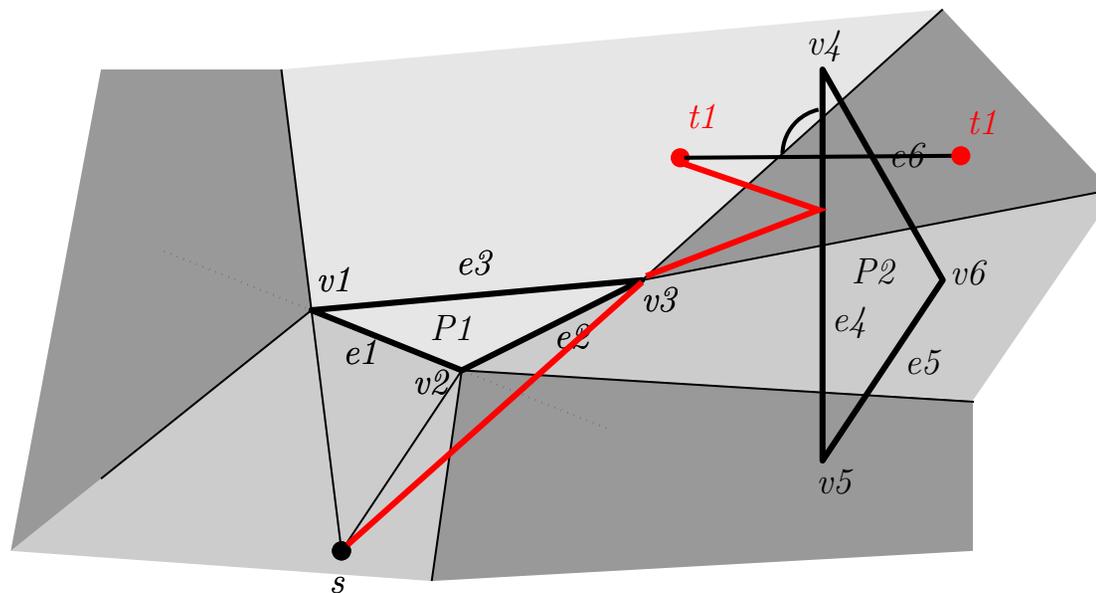
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



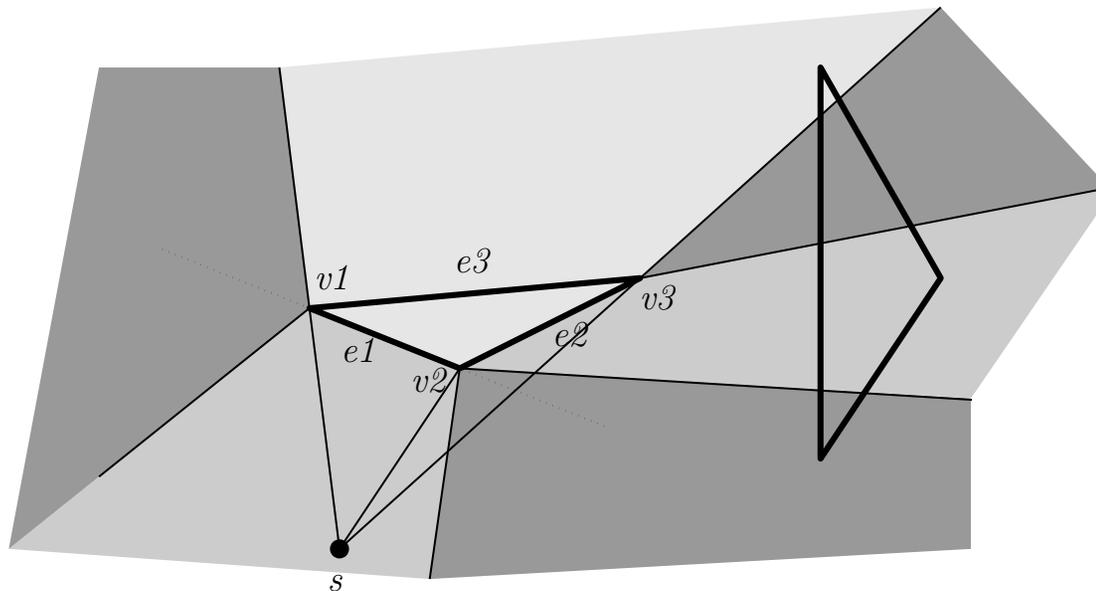
Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



Benutzung der Last Step SPM: Alg. 1.10

- Ziel t_1 : Nutze Last Step SPM von P_k
- Induktiv nutze Last Step SPM von P_{k-1}



Analyse der Query! **Lemma 1.33**

- Annahme: Alle Last Step SPM sind gegeben

-

$$\sum_{i=1}^k \log n_i \quad \text{mit} \quad \sum_{i=1}^k n_i = n$$

- Worst-Case:

$$n_i = \frac{n}{k}$$

- Alles zusammen:

$$O\left(k \log \frac{n}{k}\right)$$

Berechnung S_1, \dots, S_k : Alg. 1.11

Berechnung S_1, \dots, S_k : Alg. 1.11

- Queries: Backward

Berechnung S_1, \dots, S_k : Alg. 1.11

- Queries: Backward
- Berechnung: Forward

Berechnung S_1, \dots, S_k : Alg. 1.11

- Queries: Backward
- Berechnung: Forward
- SPM für P_1

Berechnung S_1, \dots, S_k : Alg. 1.11

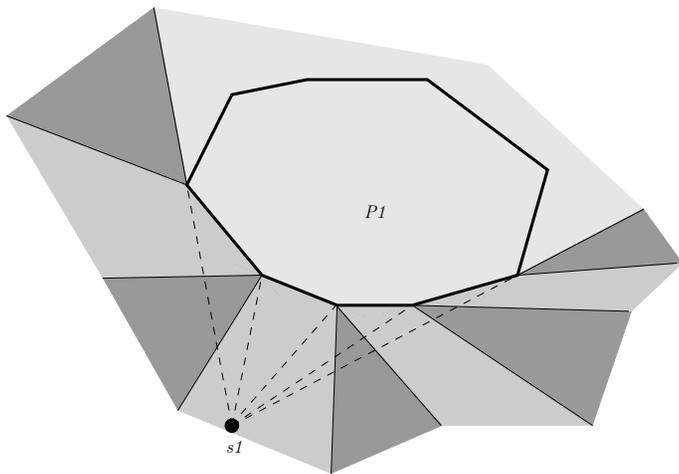
- Queries: Backward
- Berechnung: Forward
- SPM für P_1
- Sichtbare konvexe Kette/Baum der Strahlen: $O(n_1)$

Berechnung S_1, \dots, S_k : Alg. 1.11

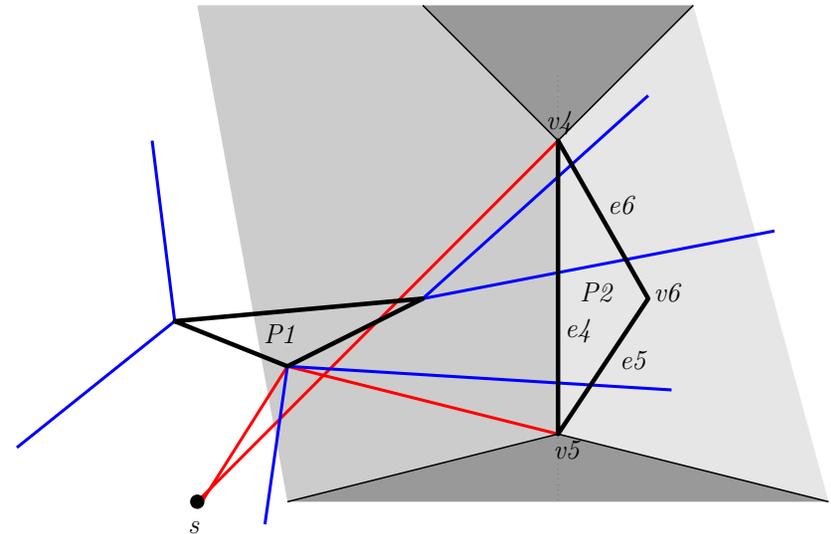
- Queries: Backward
- Berechnung: Forward
- SPM für P_1
- Sichtbare konvexe Kette/Baum der Strahlen: $O(n_1)$
- Disjunkte Reflektionen!!

Berechnung S_1, \dots, S_k : Alg. 1.11

- Queries: Backward
- Berechnung: Forward
- SPM für P_1
- Sichtbare konvexe Kette/Baum der Strahlen: $O(n_1)$
- Disjunkte Reflektionen!!

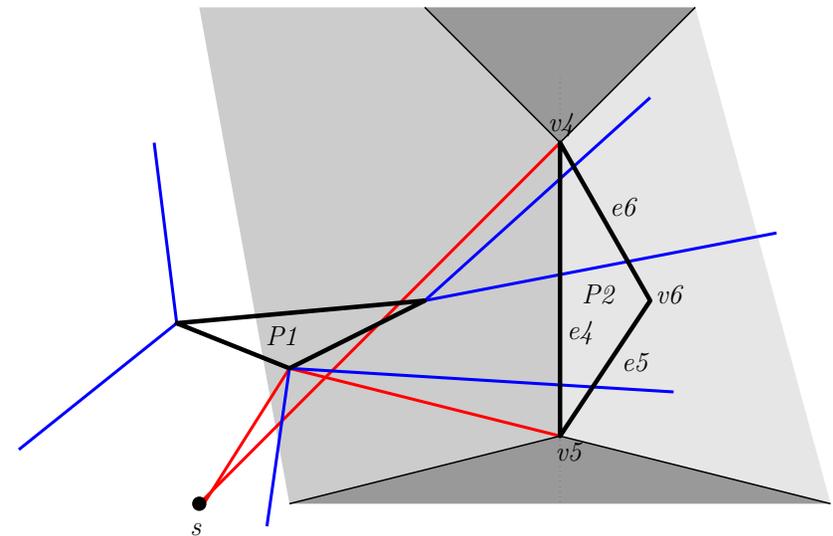


Berechnung S_1, \dots, S_k : Alg. 1.11



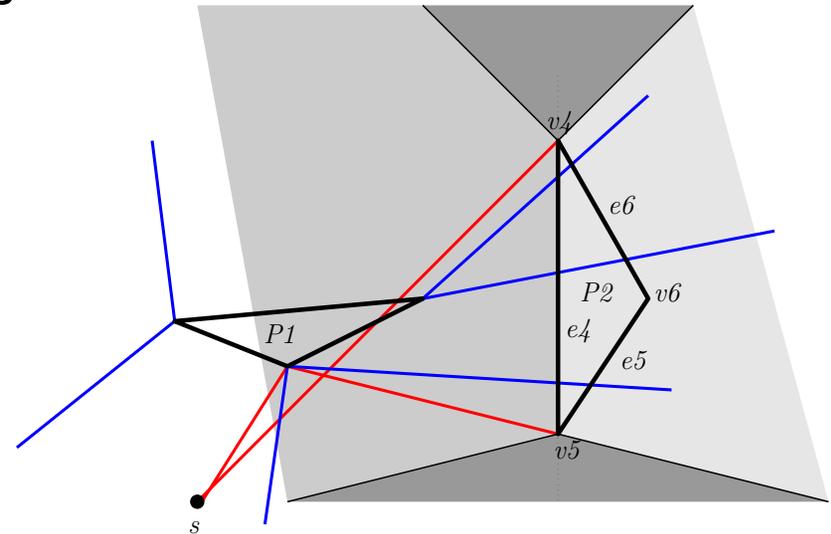
Berechnung S_1, \dots, S_k : Alg. 1.11

- SPM von P_i aus SPM of P_{i-1}, \dots, P_1



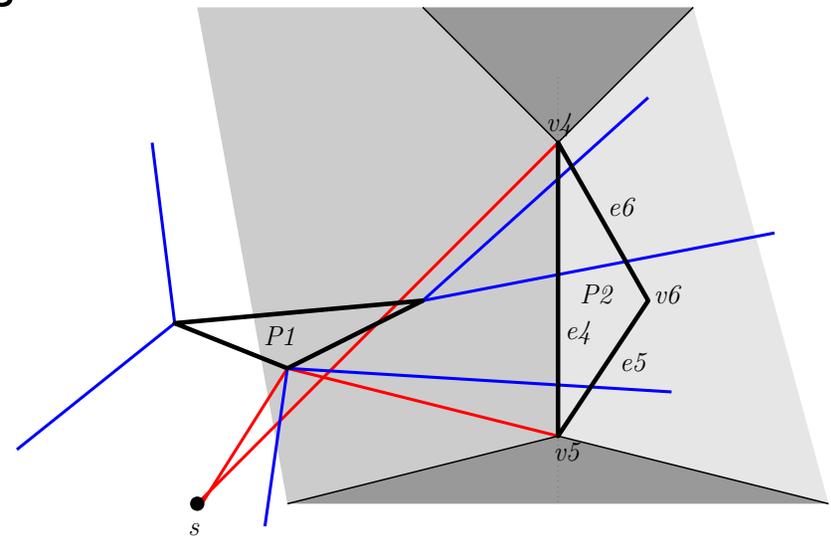
Berechnung S_1, \dots, S_k : Alg. 1.11

- SPM von P_i aus SPM of P_{i-1}, \dots, P_1
- Letztes Segment des Kürzesten Weges von s zu Knoten von P_i



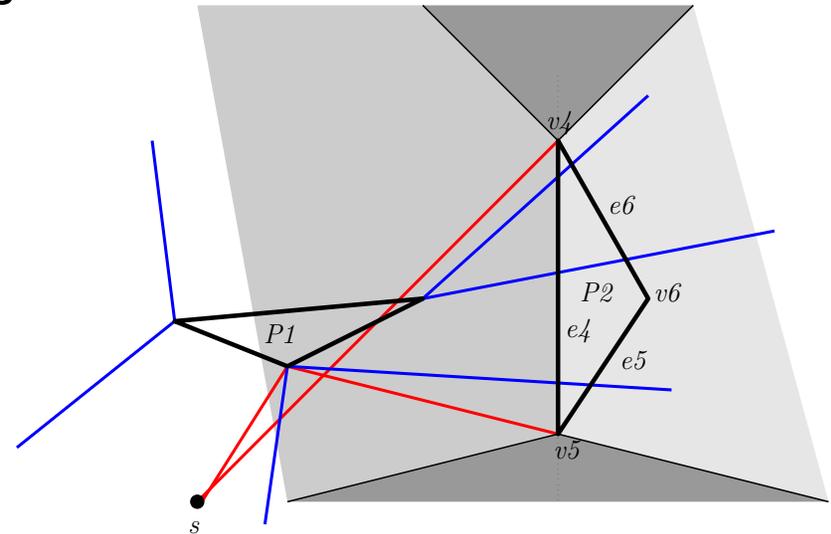
Berechnung S_1, \dots, S_k : Alg. 1.11

- SPM von P_i aus SPM of P_{i-1}, \dots, P_1
- Letztes Segment des Kürzesten Weges von s zu Knoten von P_i
- Query: Nutze SPM P_{i-1}, \dots, P_1



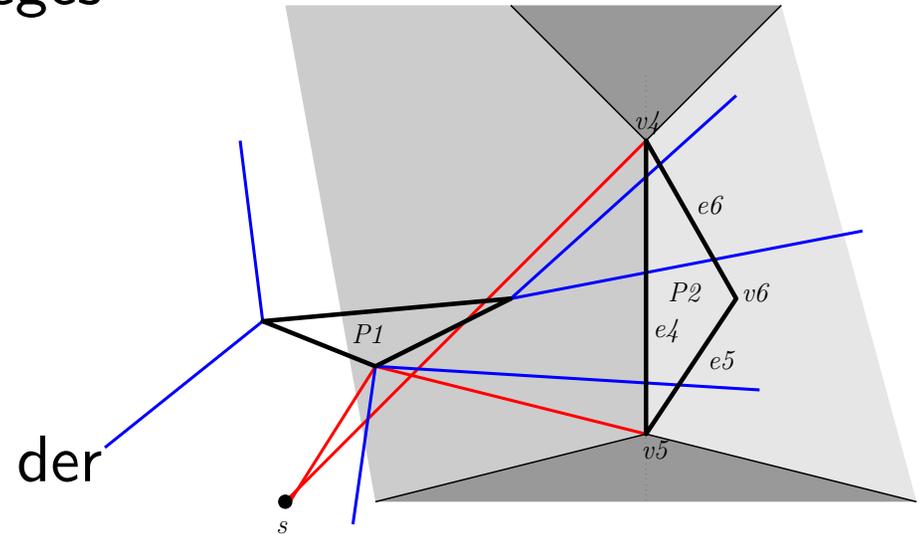
Berechnung S_1, \dots, S_k : Alg. 1.11

- SPM von P_i aus SPM of P_{i-1}, \dots, P_1
- Letztes Segment des Kürzesten Weges von s zu Knoten von P_i
- Query: Nutze SPM P_{i-1}, \dots, P_1
- Laufzeit: $O\left(n_i(i-1) \log \frac{N_{i-1}}{i-1}\right)$
mit $N_j := \sum_{l=1}^j n_l$



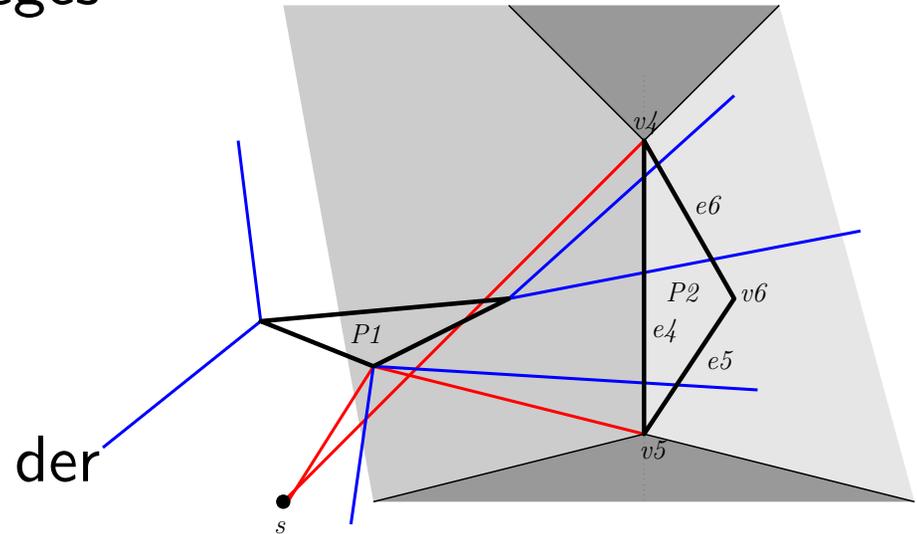
Berechnung S_1, \dots, S_k : Alg. 1.11

- SPM von P_i aus SPM of P_{i-1}, \dots, P_1
- Letztes Segment des Kürzesten Weges von s zu Knoten von P_i
- Query: Nutze SPM P_{i-1}, \dots, P_1
- Laufzeit: $O\left(n_i(i-1) \log \frac{N_{i-1}}{i-1}\right)$
mit $N_j := \sum_{l=1}^j n_l$
- Sichtbare konvexe Kette/Baum der Strahlen



Berechnung S_1, \dots, S_k : Alg. 1.11

- SPM von P_i aus SPM of P_{i-1}, \dots, P_1
- Letztes Segment des Kürzesten Weges von s zu Knoten von P_i
- Query: Nutze SPM P_{i-1}, \dots, P_1
- Laufzeit: $O\left(n_i(i-1) \log \frac{N_{i-1}}{i-1}\right)$
mit $N_j := \sum_{l=1}^j n_l$
- Sichtbare konvexe Kette/Baum der Strahlen
- Disjunkt wegen konvexer Kette!!



Analyse Berechnung S_1, \dots, S_k : **Theorem 1.34**

Analyse Berechnung S_1, \dots, S_k : **Theorem 1.34**

Rekursiv: P_2, \dots, P_k

Analyse Berechnung S_1, \dots, S_k : **Theorem 1.34**

Rekursiv: P_2, \dots, P_k

Gesamtlaufzeit:

Analyse Berechnung S_1, \dots, S_k : **Theorem 1.34**

Rekursiv: P_2, \dots, P_k

Gesamtlaufzeit:

$$\sum_{i=2}^k n_i (i-1) \log \frac{N_{i-1}}{i-1}$$

Analyse Berechnung S_1, \dots, S_k : **Theorem 1.34**

Rekursiv: P_2, \dots, P_k

Gesamtlaufzeit:

$$\sum_{i=2}^k n_i (i-1) \log \frac{N_{i-1}}{i-1}$$

$$n_i (i-1) \log \frac{N_{i-1}}{i-1} \leq n_i k \log \frac{n}{k}$$

Analyse Berechnung S_1, \dots, S_k : **Theorem 1.34**

Rekursiv: P_2, \dots, P_k

Gesamtlaufzeit:

$$\sum_{i=2}^k n_i(i-1) \log \frac{N_{i-1}}{i-1}$$

$$n_i(i-1) \log \frac{N_{i-1}}{i-1} \leq n_i k \log \frac{n}{k}$$

Gesamtlaufzeit:

Analyse Berechnung S_1, \dots, S_k : **Theorem 1.34**

Rekursiv: P_2, \dots, P_k

Gesamtlaufzeit:

$$\sum_{i=2}^k n_i (i-1) \log \frac{N_{i-1}}{i-1}$$

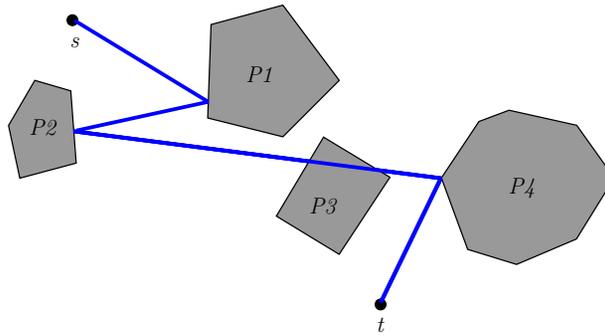
$$n_i (i-1) \log \frac{N_{i-1}}{i-1} \leq n_i k \log \frac{n}{k}$$

Gesamtlaufzeit:

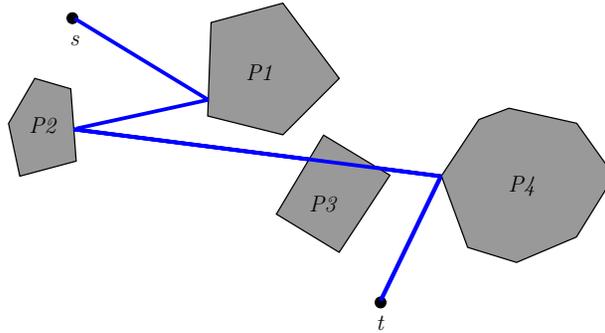
$$O\left(kn \log \frac{n}{k}\right)!$$

Zusammenfassung!!

Zusammenfassung!!



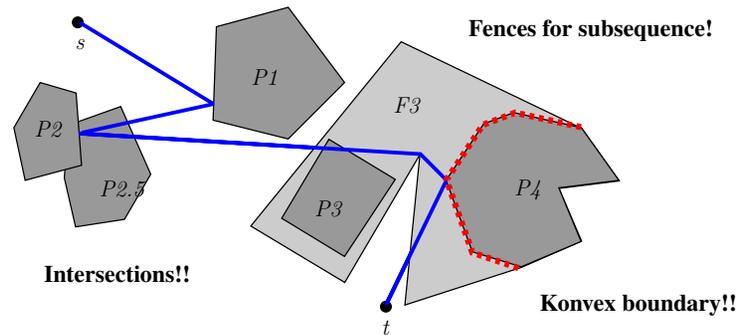
Zusammenfassung!!



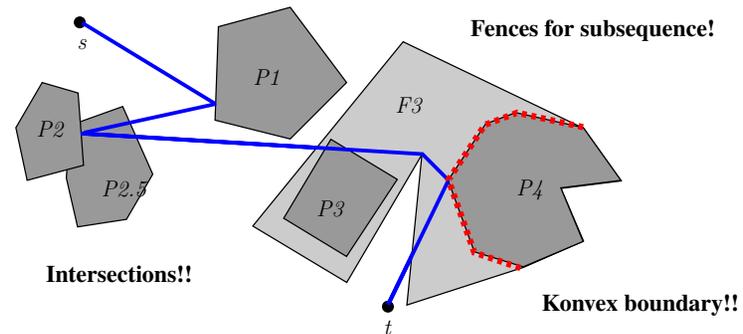
- Einfache Version:
- Disjunkte, konvexe Polygone, keine Zäune
- $O(nk \log \frac{n}{k})$
- Build(Query): $O(nk \log \frac{n}{k})$
- Komplexität: $O(n)$
- Query (festes s): $O(k \log \frac{n}{k})$
- **Theorem 1.34**

Erweiterung!

Erweiterung!

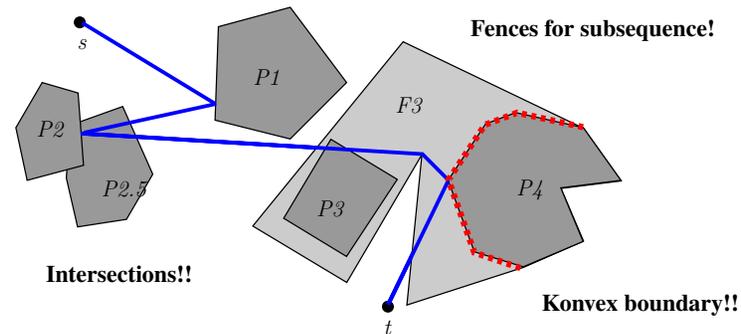


Erweiterung!



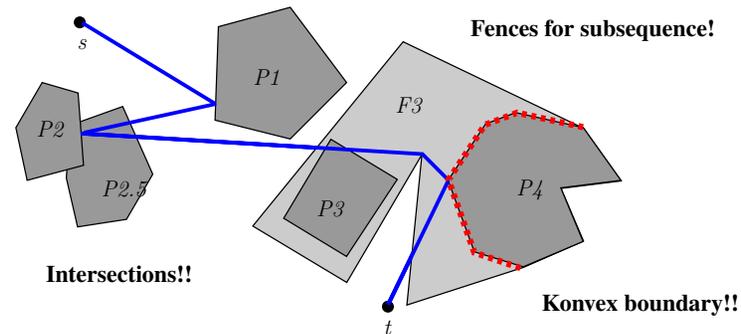
- Komplexe Version:

Erweiterung!



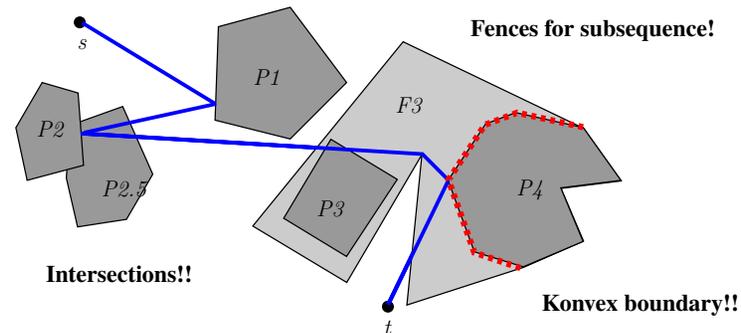
- Komplexe Version:
- Nicht-disjunkte, konvexe Polygone, Zäune

Erweiterung!



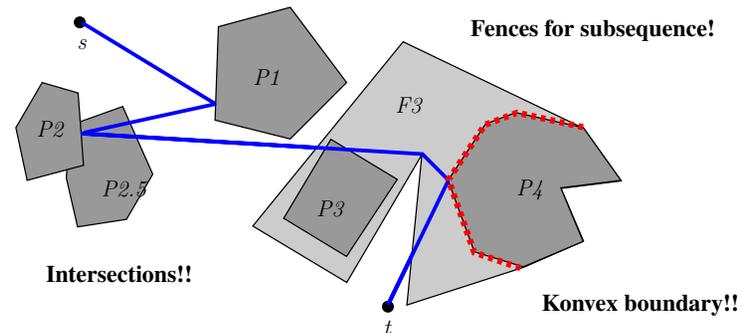
- Komplexe Version:
- Nicht-disjunkte, konvexe Polygone, Zäune
- $O(nk^2 \log n)$ insgesamt

Erweiterung!



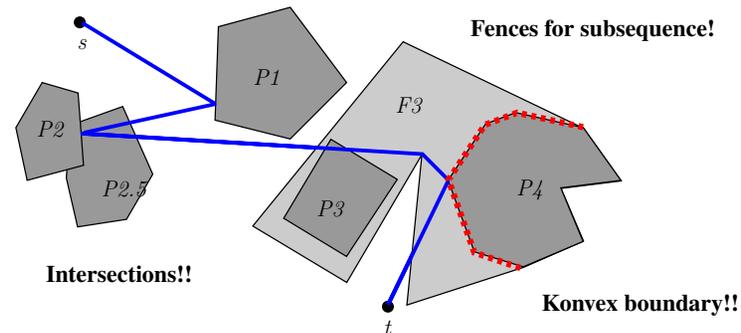
- Komplexe Version:
- Nicht-disjunkte, konvexe Polygone, Zäune
- $O(nk^2 \log n)$ insgesamt
- Build(Query): $O(nk^2 \log n)$

Erweiterung!



- Komplexe Version:
- Nicht-disjunkte, konvexe Polygone, Zäune
- $O(nk^2 \log n)$ insgesamt
- Build(Query): $O(nk^2 \log n)$
- Komplexität: $O(kn)$

Erweiterung!



- Komplexe Version:
- Nicht-disjunkte, konvexe Polygone, Zäune
- $O(nk^2 \log n)$ insgesamt
- Build(Query): $O(nk^2 \log n)$
- Komplexität: $O(kn)$
- Query (festes s): $O(kn)$

- Lemma 1.35: Reflexionsbereich ist Baum

- Lemma 1.35: Reflexionsbereich ist Baum
- Theorem 1.36/Theorem 1.37

Polyeder-Szene in 3D

Polyeder-Szene in 3D

- Startpunkt s ,

•
 s

Polyeder-Szene in 3D

- Startpunkt s , Zielpunkt t



s



t

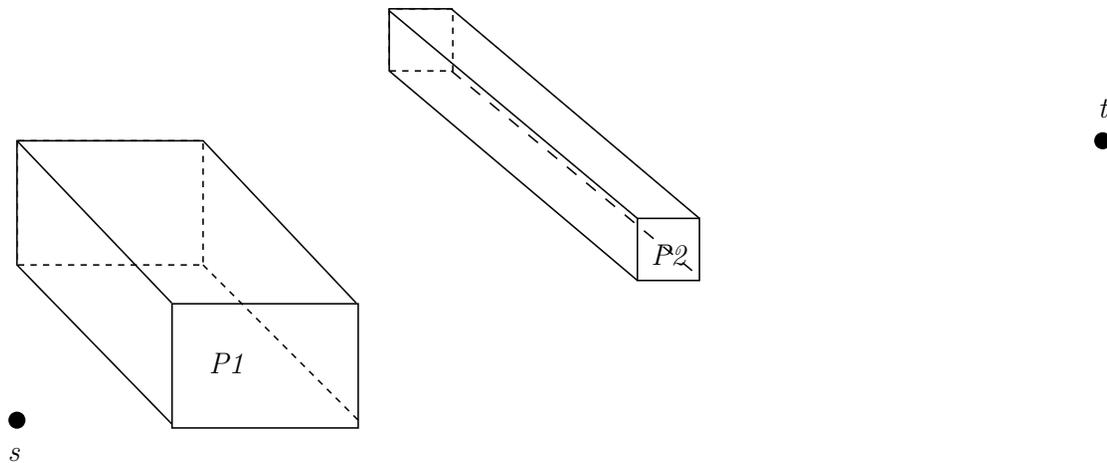
Polyeder-Szene in 3D

- Startpunkt s , Zielpunkt t
- Menge von Polyedern



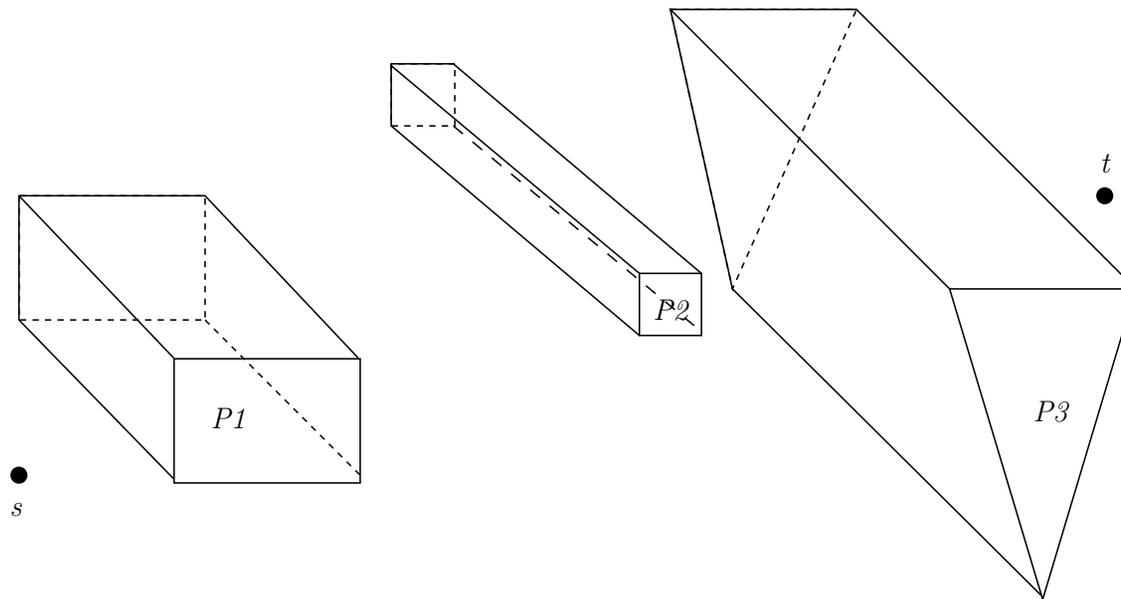
Polyeder-Szene in 3D

- Startpunkt s , Zielpunkt t
- Menge von Polyedern



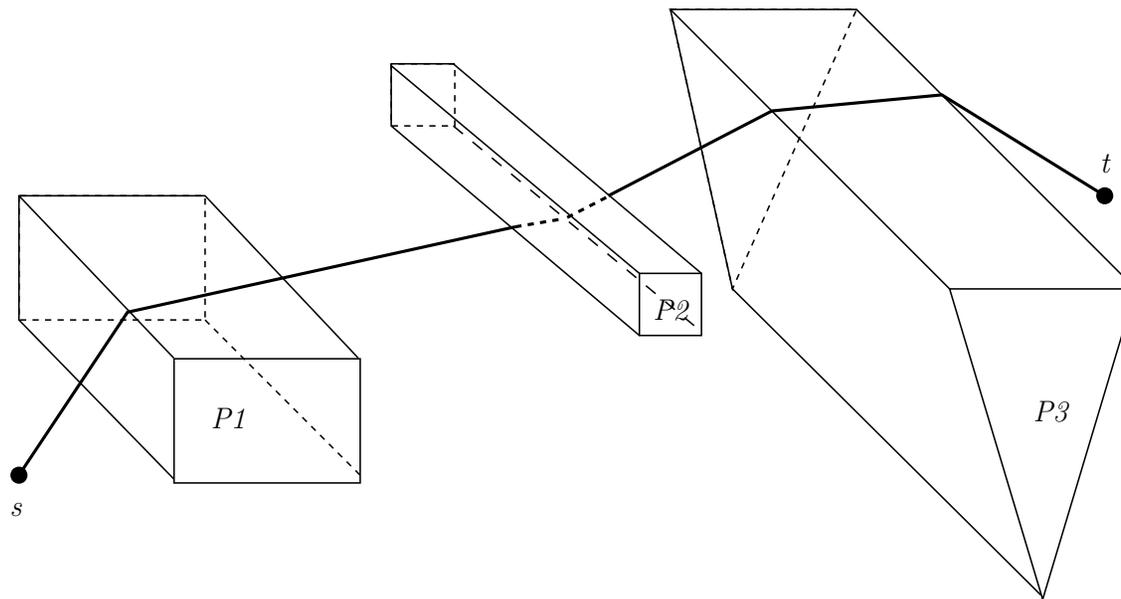
Polyeder-Szene in 3D

- Startpunkt s , Zielpunkt t
- Menge von Polyedern



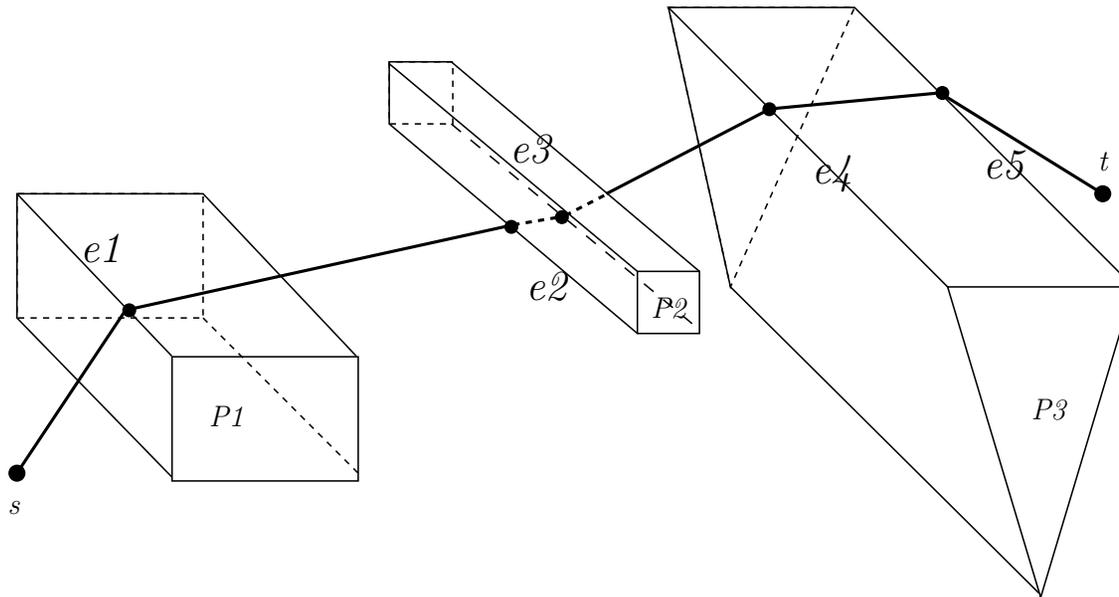
Polyeder-Szene in 3D

- Startpunkt s , Zielpunkt t
- Menge von Polyedern
- Kürzester Weg von s nach t :



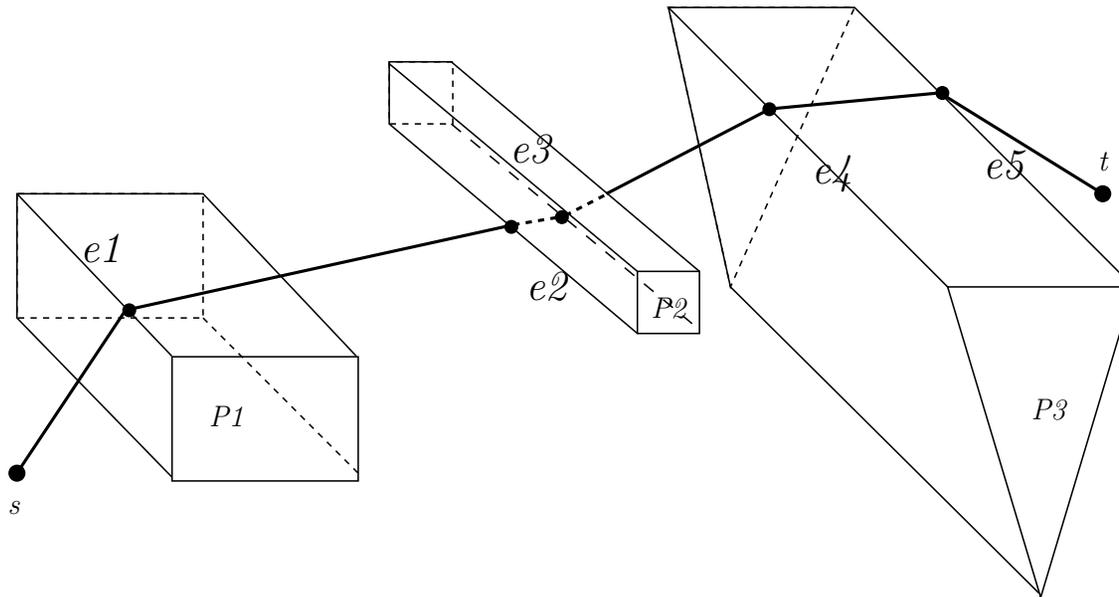
Polyeder-Szene in 3D

- Startpunkt s , Zielpunkt t
- Menge von Polyedern
- Kürzester Weg von s nach t :

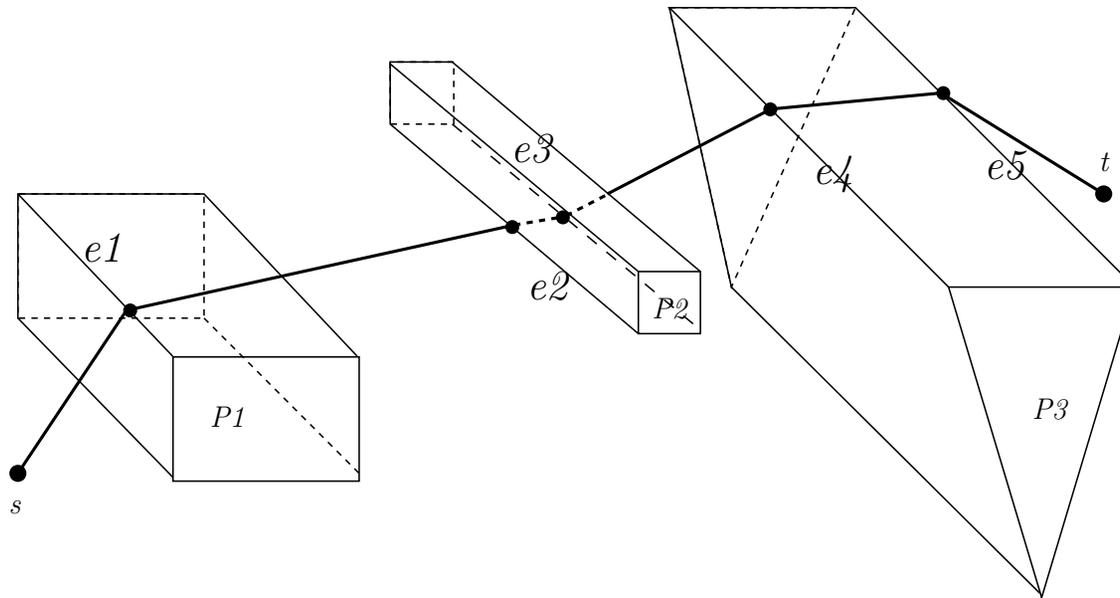


Polyeder-Szene in 3D

- Startpunkt s , Zielpunkt t
- Menge von Polyedern
- Kürzester Weg von s nach t : NP-hard

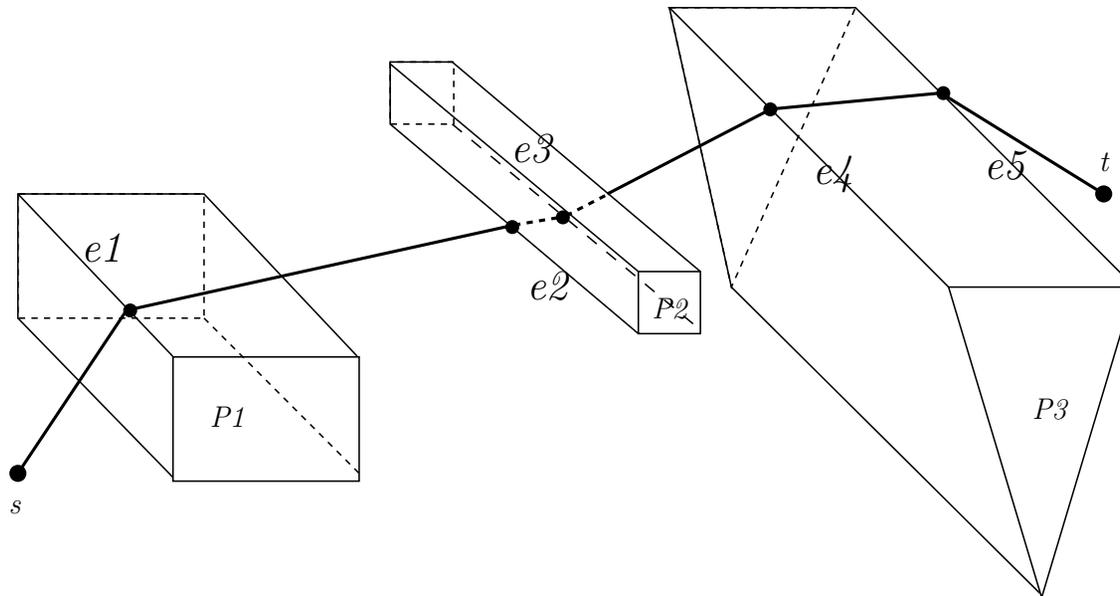


Polyeder-Szene in 3D



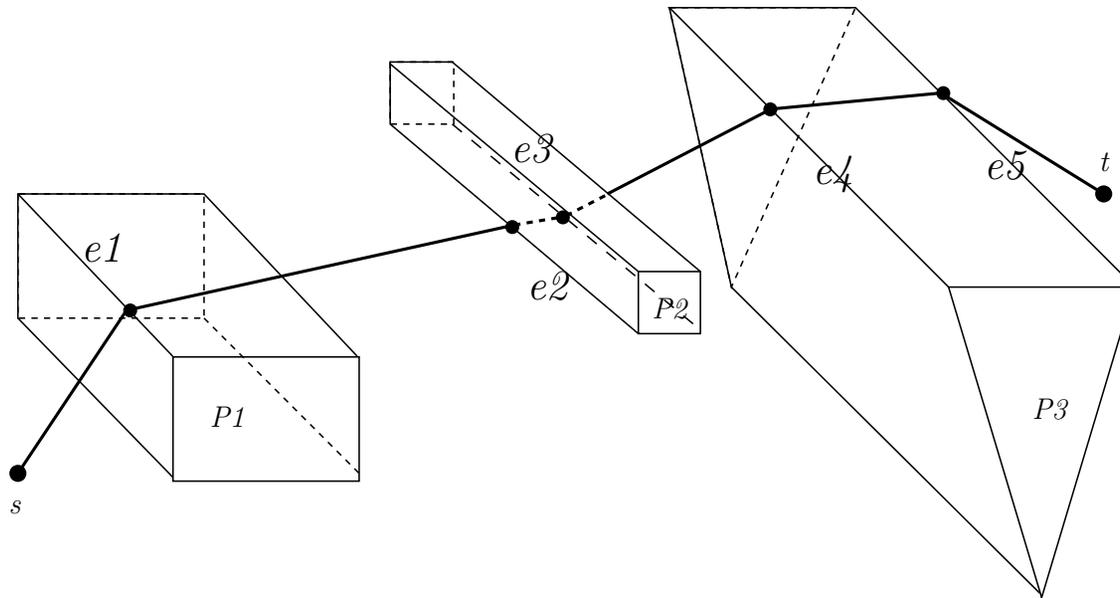
Polyeder-Szene in 3D

- Teilprobleme



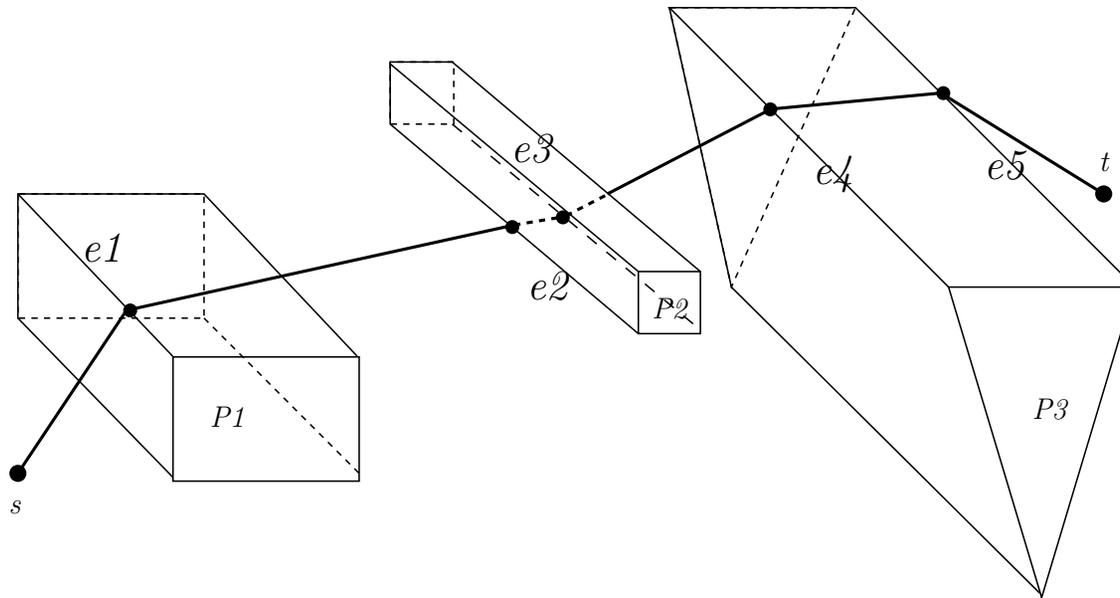
Polyeder-Szene in 3D

- Teilprobleme
- 1) Kantenreihenfolge



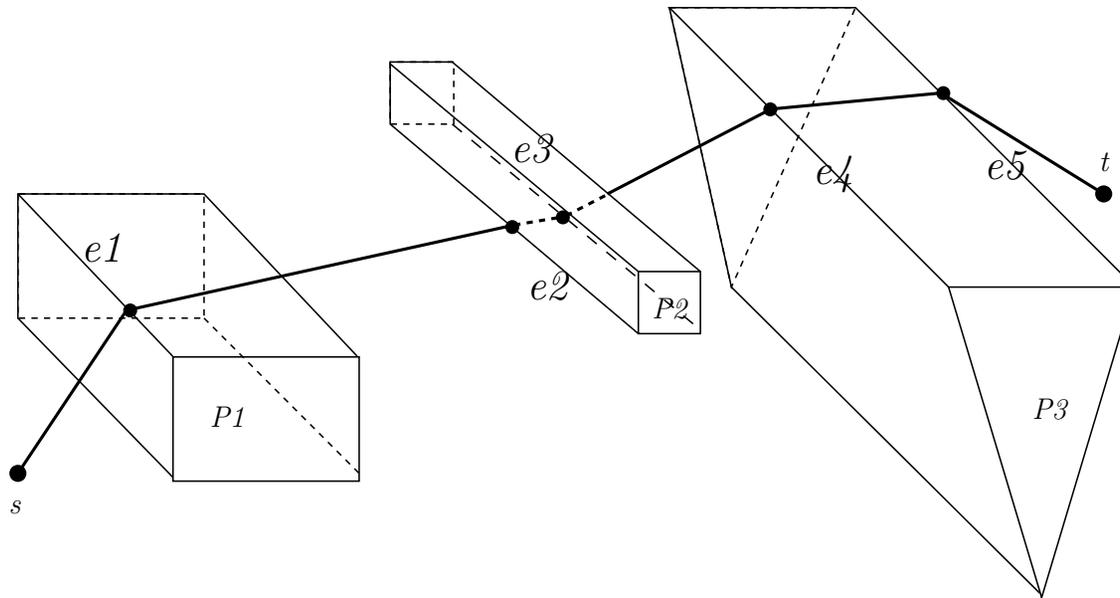
Polyeder-Szene in 3D

- Teilprobleme
- 1) Kantenreihenfolge
- 2) Verschiebung auf der Kante



Polyeder-Szene in 3D

- Teilprobleme
- 1) Kantenreihenfolge
- 2) Verschiebung auf der Kante
- Bereits 1) ist NP hard



Kantenreihenfolge: NP hard

Kantenreihenfolge: NP hard

- Entscheidungsproblem,

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke,

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega$:

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega: x \in \Omega:$

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega: x \in \Omega: \text{gilt } x \in S?$

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega$: $x \in \Omega$: gilt $x \in S$?
- Problem S ist NP -vollständig

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega$: $x \in \Omega$: gilt $x \in S$?
- Problem S ist NP -vollständig

1. Liegt in NP

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega$: $x \in \Omega$: gilt $x \in S$?
- Problem S ist NP -vollständig
 1. Liegt in NP
 2. Jedes andere Problem $S' \in NP$ läßt sich auf S in polynomieller Zeit auf S reduzieren

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega$: $x \in \Omega$: gilt $x \in S$?
- Problem S ist NP -vollständig
 1. Liegt in NP
 2. Jedes andere Problem $S' \in NP$ läßt sich auf S in polynomieller Zeit auf S reduzieren
- NP hard: Nur zweiter Teil (Optimierungsprobleme)

Kantenreihenfolge: NP hard

- Entscheidungsproblem, E-Problem mit Schranke, Optimierungsproblem
- $S \subset \Omega: x \in \Omega: \text{ gilt } x \in S?$
- Problem S ist NP -vollständig
 1. Liegt in NP
 2. Jedes andere Problem $S' \in NP$ läßt sich auf S in polynomieller Zeit auf S reduzieren
- NP hard: Nur zweiter Teil (Optimierungsprobleme)
- 3-SAT ist NP vollständig (Cook)

Kantenreihenfolge: NP hard

Kantenreihenfolge: NP hard

- Reduktion $S' \subset \Omega'$ auf $S \subset \Omega$

Kantenreihenfolge: NP hard

- Reduktion $S' \subset \Omega'$ auf $S \subset \Omega$
- Funktion $f : \Omega' \rightarrow \Omega$

Kantenreihenfolge: NP hard

- Reduktion $S' \subset \Omega'$ auf $S \subset \Omega$
- Funktion $f : \Omega' \rightarrow \Omega$
 1. $\forall x' \in \Omega'$: $f(x')$ in polynomieller Zeit ($|x'|$)
 2. $\forall x' \in \Omega'$: $f(x') \in S \Leftrightarrow x' \in S'$

Kantenreihenfolge: NP hard

- Reduktion $S' \subset \Omega'$ auf $S \subset \Omega$
- Funktion $f : \Omega' \rightarrow \Omega$
 1. $\forall x' \in \Omega'$: $f(x')$ in polynomieller Zeit ($|x'|$)
 2. $\forall x' \in \Omega'$: $f(x') \in S \Leftrightarrow x' \in S'$
- 3-SAT NP vollständig

Kantenreihenfolge: NP hard

- Reduktion $S' \subset \Omega'$ auf $S \subset \Omega$
- Funktion $f : \Omega' \rightarrow \Omega$
 1. $\forall x' \in \Omega'$: $f(x')$ in polynomieller Zeit ($|x'|$)
 2. $\forall x' \in \Omega'$: $f(x') \in S \Leftrightarrow x' \in S'$
- 3-SAT NP vollständig
- 3-SAT reduzieren auf Kantenreihenfolge

3-SAT reduzieren auf Kantenreihenfolge

3-SAT reduzieren auf Kantenreihenfolge

$$\alpha = \bigwedge_{i=1}^m (L_{i_1} \vee L_{i_2} \vee L_{i_3}) \text{ mit } L_{i_j} \in \{X_k, \neg X_k\}$$

3-SAT reduzieren auf Kantenreihenfolge

$$\alpha = \bigwedge_{i=1}^m (L_{i_1} \vee L_{i_2} \vee L_{i_3}) \text{ mit } L_{i_j} \in \{X_k, \neg X_k\}$$

m Klauseln mit n Variablen:

3-SAT reduzieren auf Kantenreihenfolge

$$\alpha = \bigwedge_{i=1}^m (L_{i_1} \vee L_{i_2} \vee L_{i_3}) \text{ mit } L_{i_j} \in \{X_k, \neg X_k\}$$

m Klauseln mit n Variablen: Erfüllbarkeit?

3-SAT reduzieren auf Kantenreihenfolge

$$\alpha = \bigwedge_{i=1}^m (L_{i_1} \vee L_{i_2} \vee L_{i_3}) \text{ mit } L_{i_j} \in \{X_k, \neg X_k\}$$

m Klauseln mit n Variablen: Erfüllbarkeit?

Konstruiere Parcours P_α , so dass:

3-SAT reduzieren auf Kantenreihenfolge

$$\alpha = \bigwedge_{i=1}^m (L_{i_1} \vee L_{i_2} \vee L_{i_3}) \text{ mit } L_{i_j} \in \{X_k, \neg X_k\}$$

m Klauseln mit n Variablen: Erfüllbarkeit?

Konstruiere Parcours P_α , so dass:

- Kürzester Weg (Kantenfolge) von s nach t erzeugt Belegung w

3-SAT reduzieren auf Kantenreihenfolge

$$\alpha = \bigwedge_{i=1}^m (L_{i_1} \vee L_{i_2} \vee L_{i_3}) \text{ mit } L_{i_j} \in \{X_k, \neg X_k\}$$

m Klauseln mit n Variablen: Erfüllbarkeit?

Konstruiere Parcours P_α , so dass:

- Kürzester Weg (Kantenfolge) von s nach t erzeugt Belegung w
- w erfüllt $\alpha \Rightarrow$ fertig!

3-SAT reduzieren auf Kantenreihenfolge

$$\alpha = \bigwedge_{i=1}^m (L_{i_1} \vee L_{i_2} \vee L_{i_3}) \text{ mit } L_{i_j} \in \{X_k, \neg X_k\}$$

m Klauseln mit n Variablen: Erfüllbarkeit?

Konstruiere Parcours P_α , so dass:

- Kürzester Weg (Kantenfolge) von s nach t erzeugt Belegung w
- w erfüllt $\alpha \Rightarrow$ fertig!
- w erfüllt α nicht \Rightarrow kein w erfüllt α

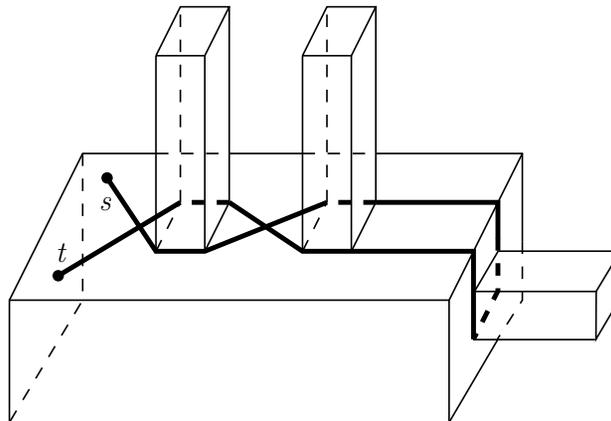
Parcours in $O(p(mn))$ erzeugen

Parcours in $O(p(mn))$ erzeugen

- 2^n Belegungen der n Variablen

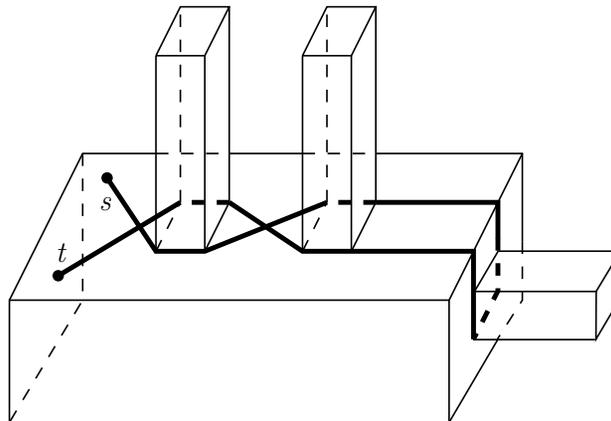
Parcours in $O(p(mn))$ erzeugen

- 2^n Belegungen der n Variablen
- 2^n *geodätisch* kürzeste Wege



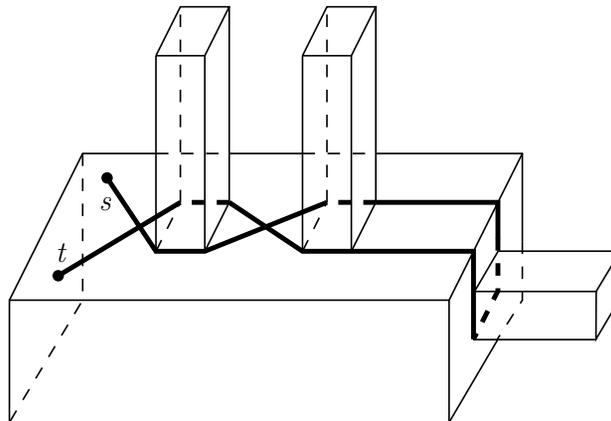
Parcours in $O(p(mn))$ erzeugen

- 2^n Belegungen der n Variablen
- 2^n *geodätisch* kürzeste Wege
- Eine davon wird die kürzeste sein



Parcours in $O(p(mn))$ erzeugen

- 2^n Belegungen der n Variablen
- 2^n *geodätisch* kürzeste Wege
- Eine davon wird die kürzeste sein
- Ergibt Variablen-Belegung nach Kantenreihenfolge



Parcours erzeugen: Prinzip

Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$

Parcours erzeugen: Prinzip

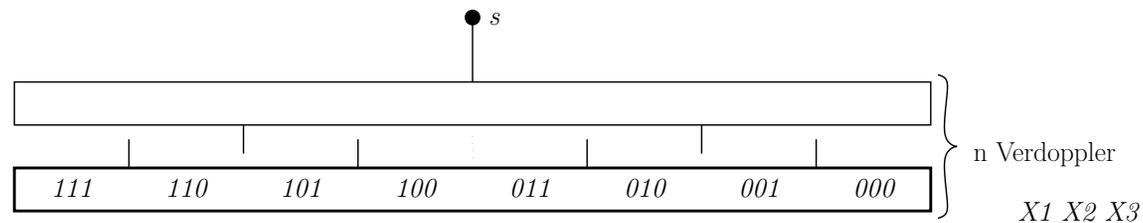
Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$

•^s
|

|
•^t

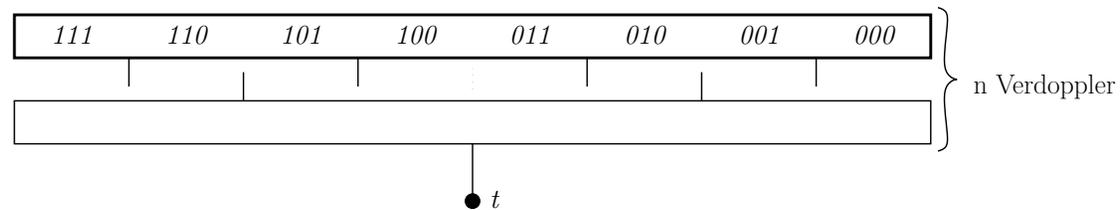
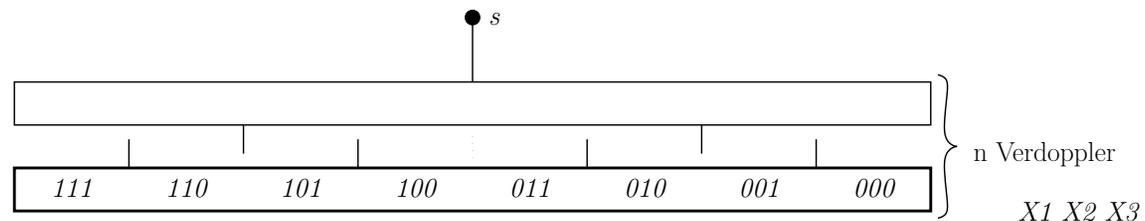
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



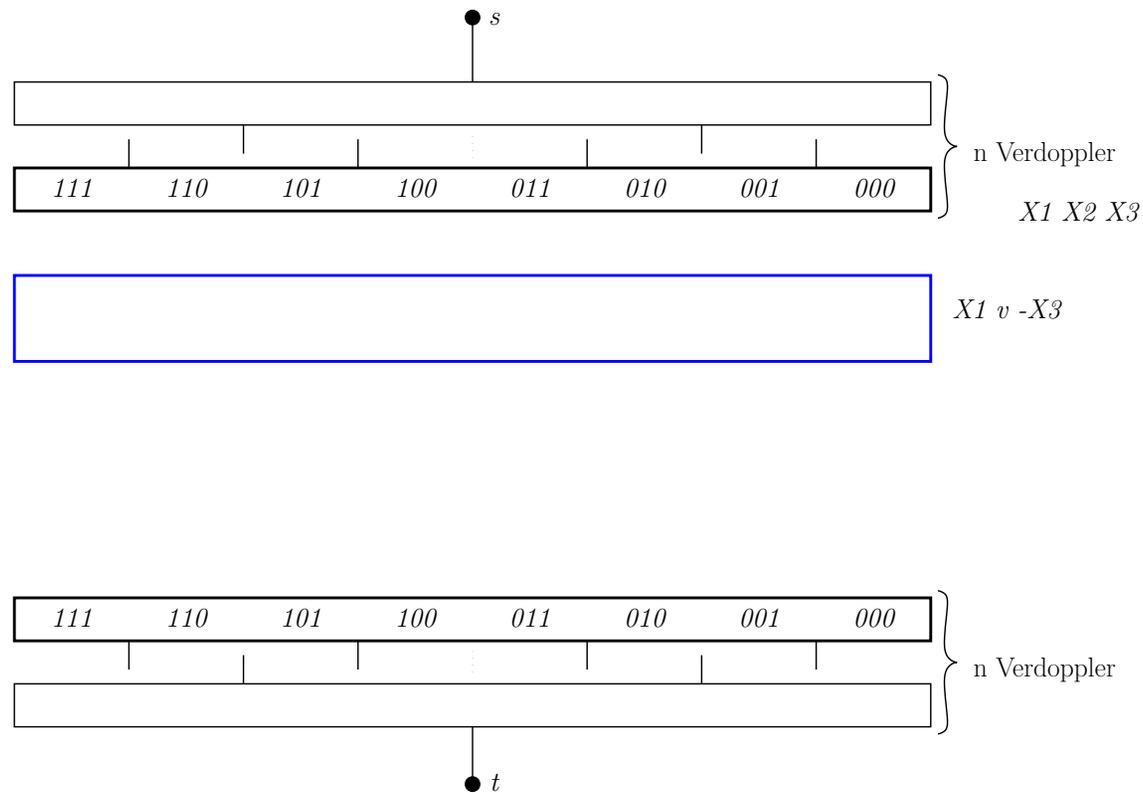
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



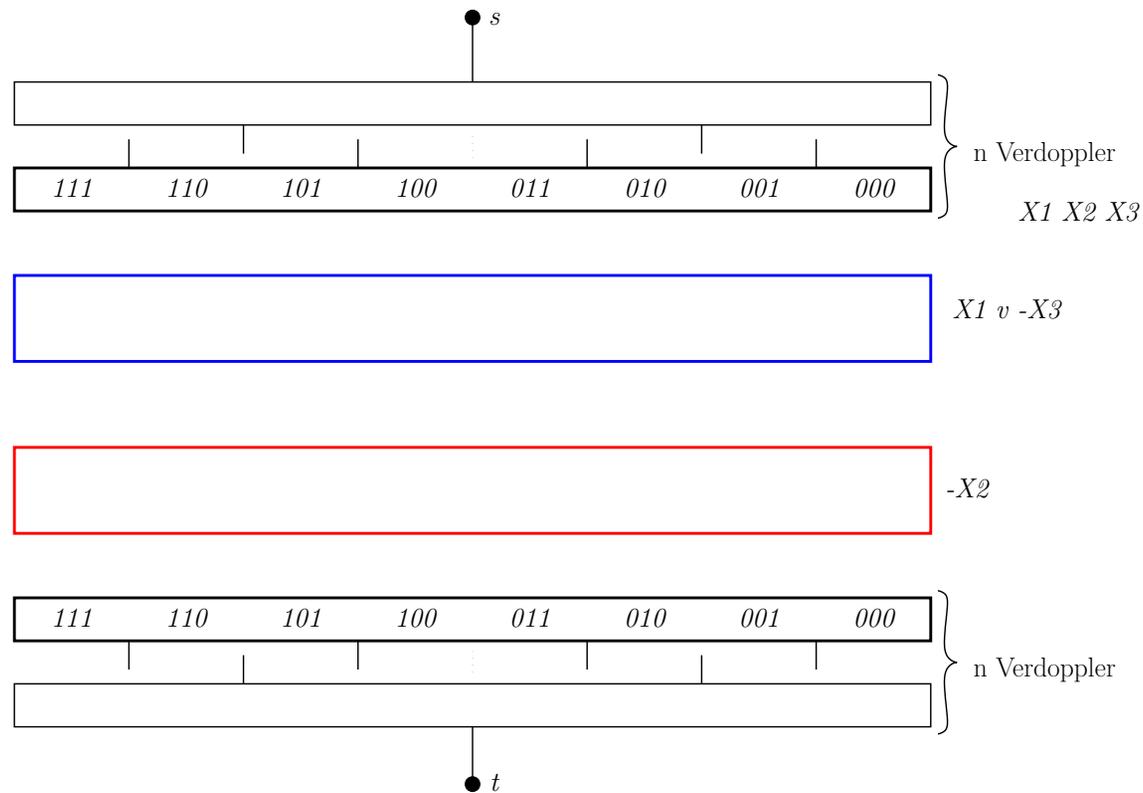
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



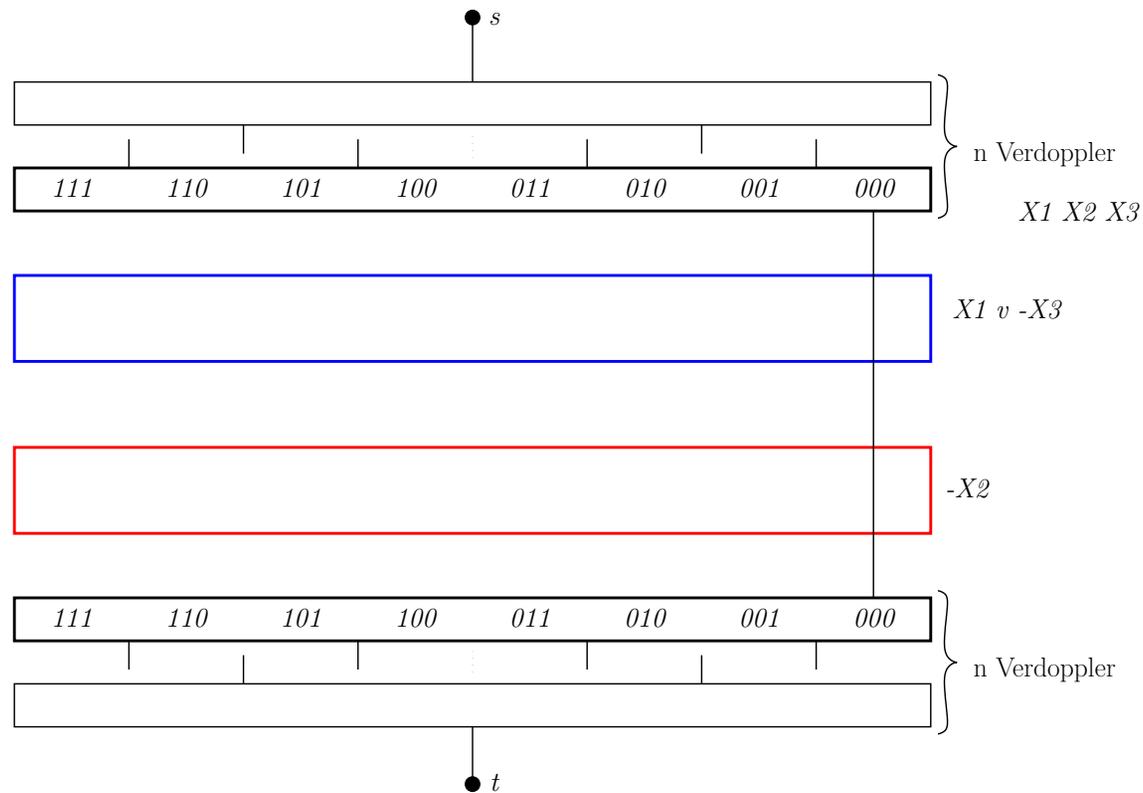
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



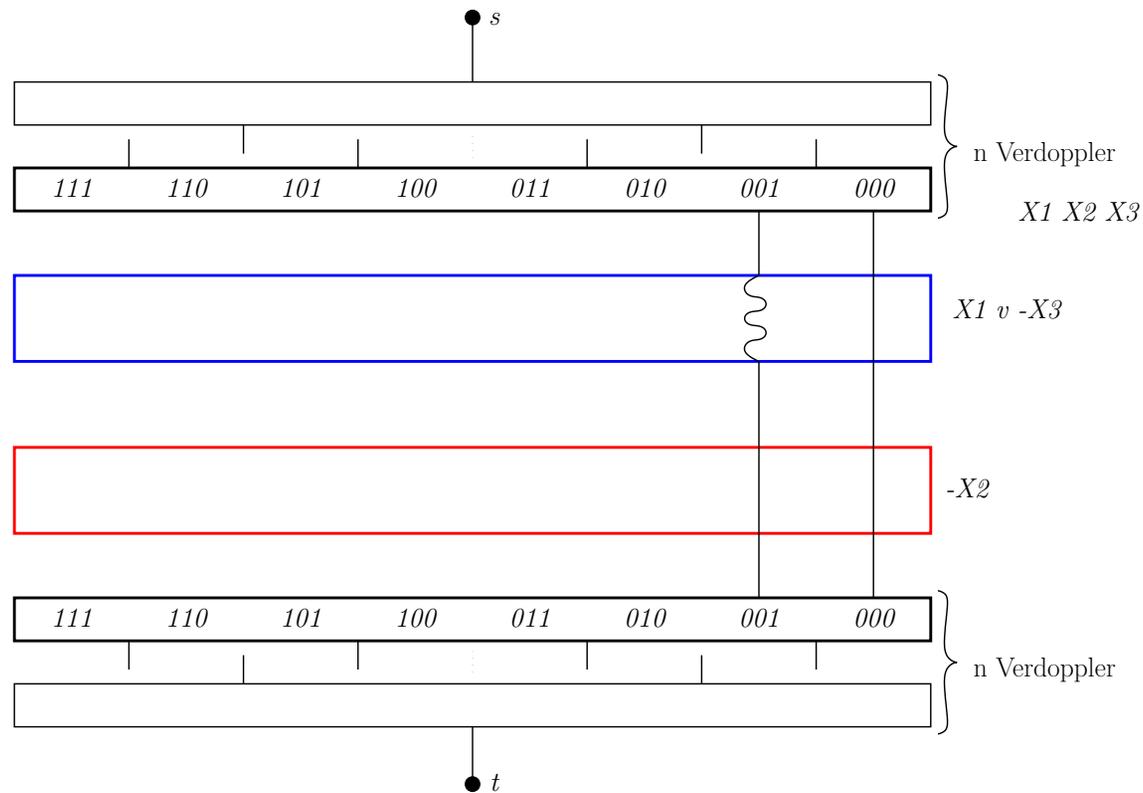
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



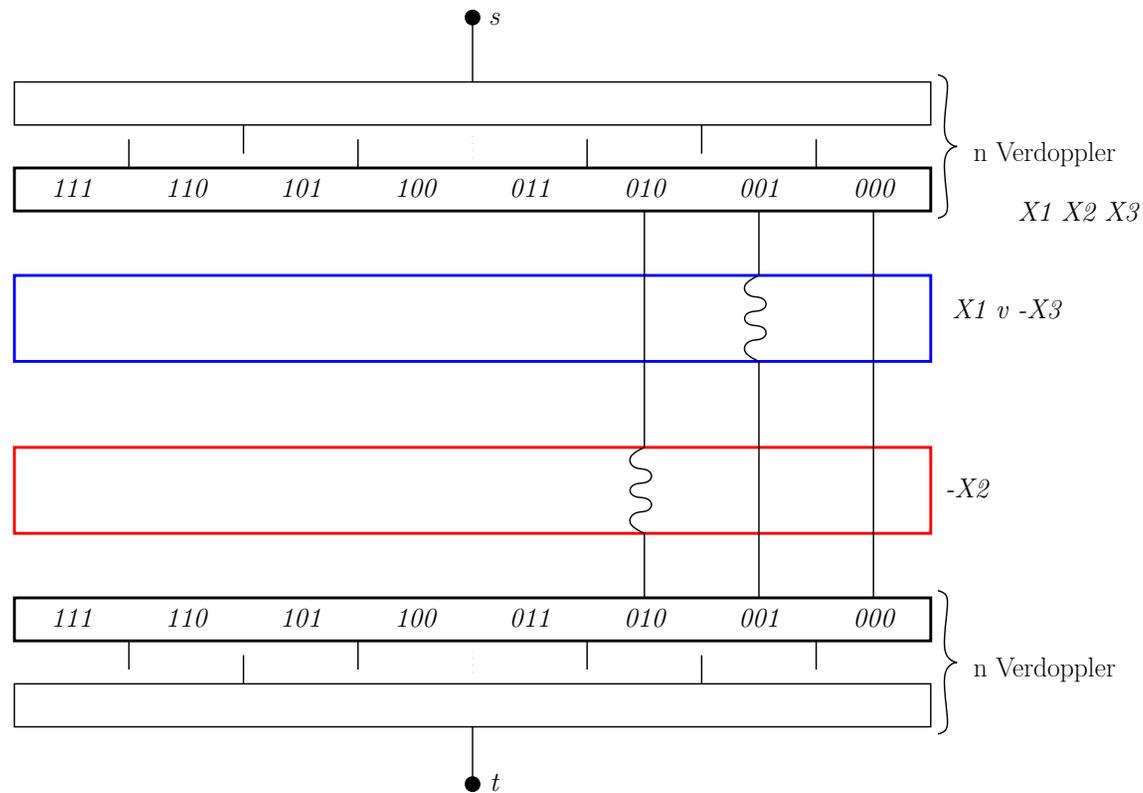
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



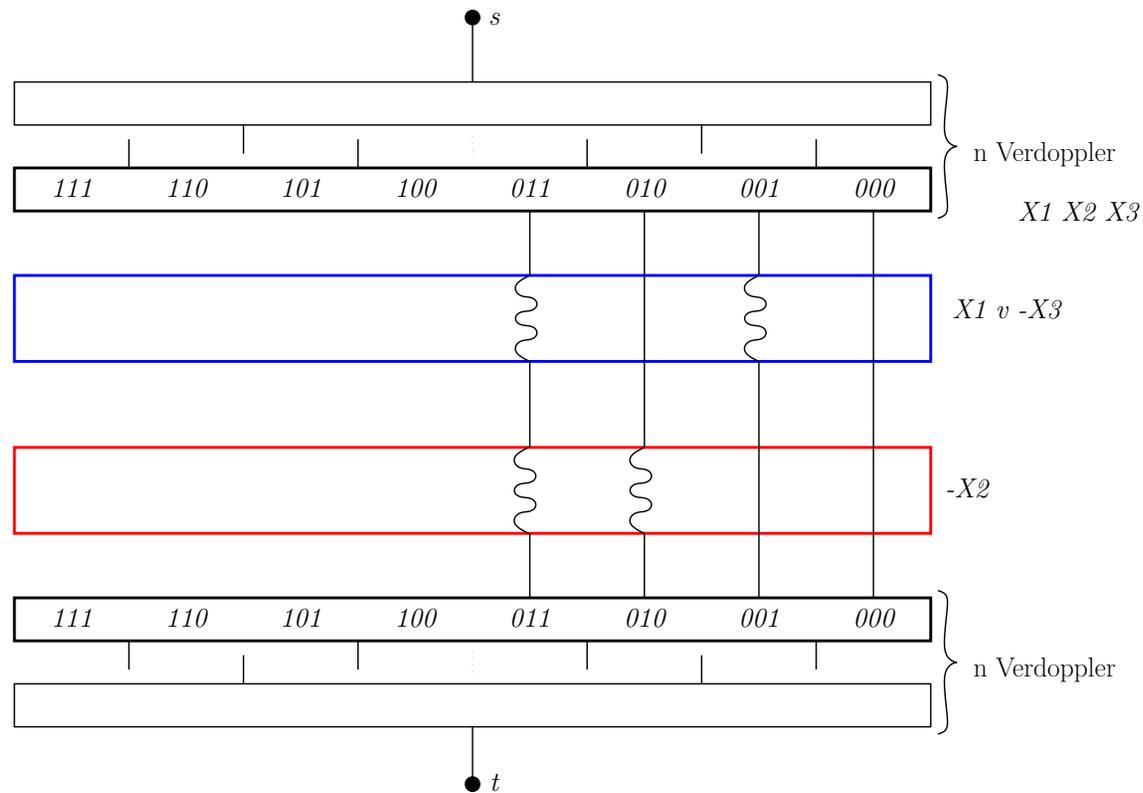
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



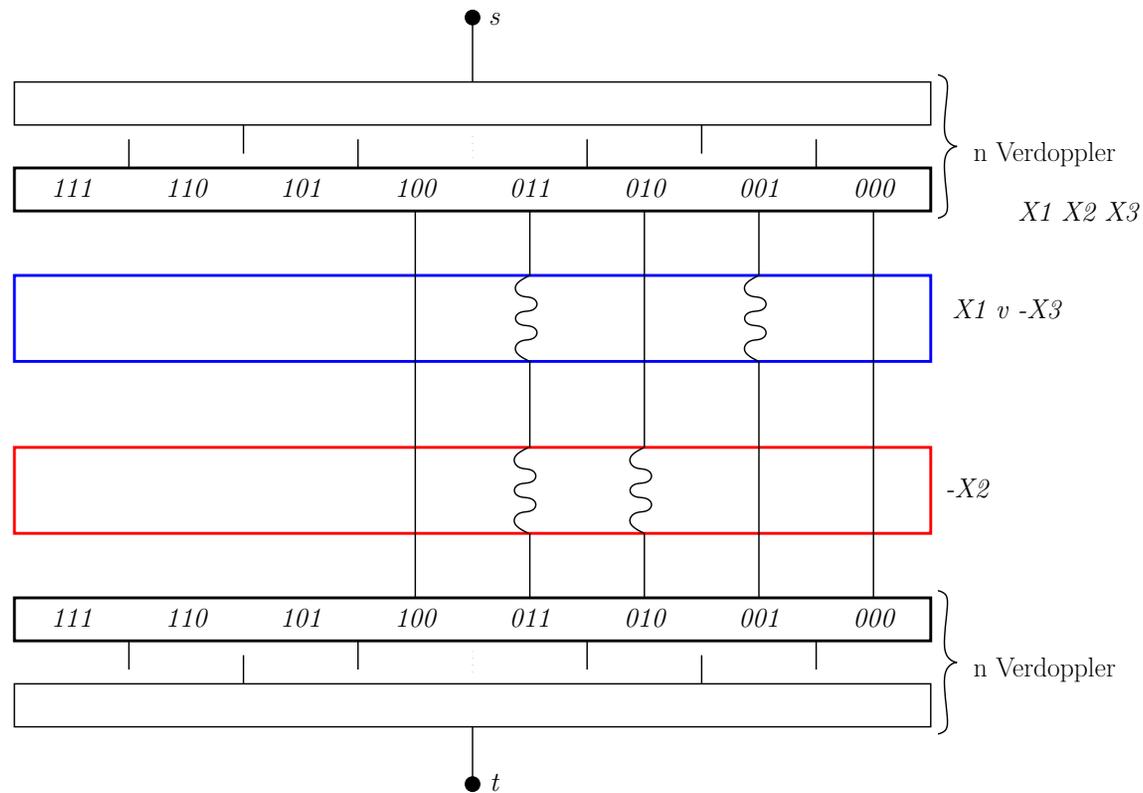
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



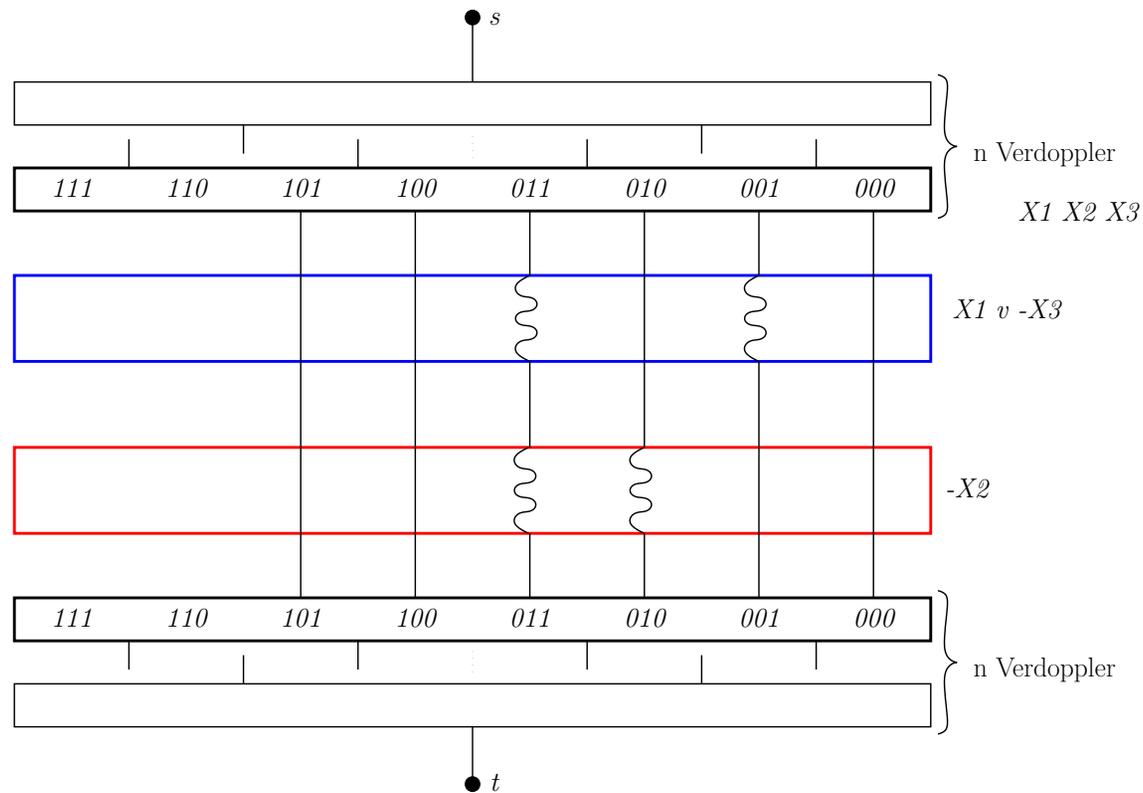
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



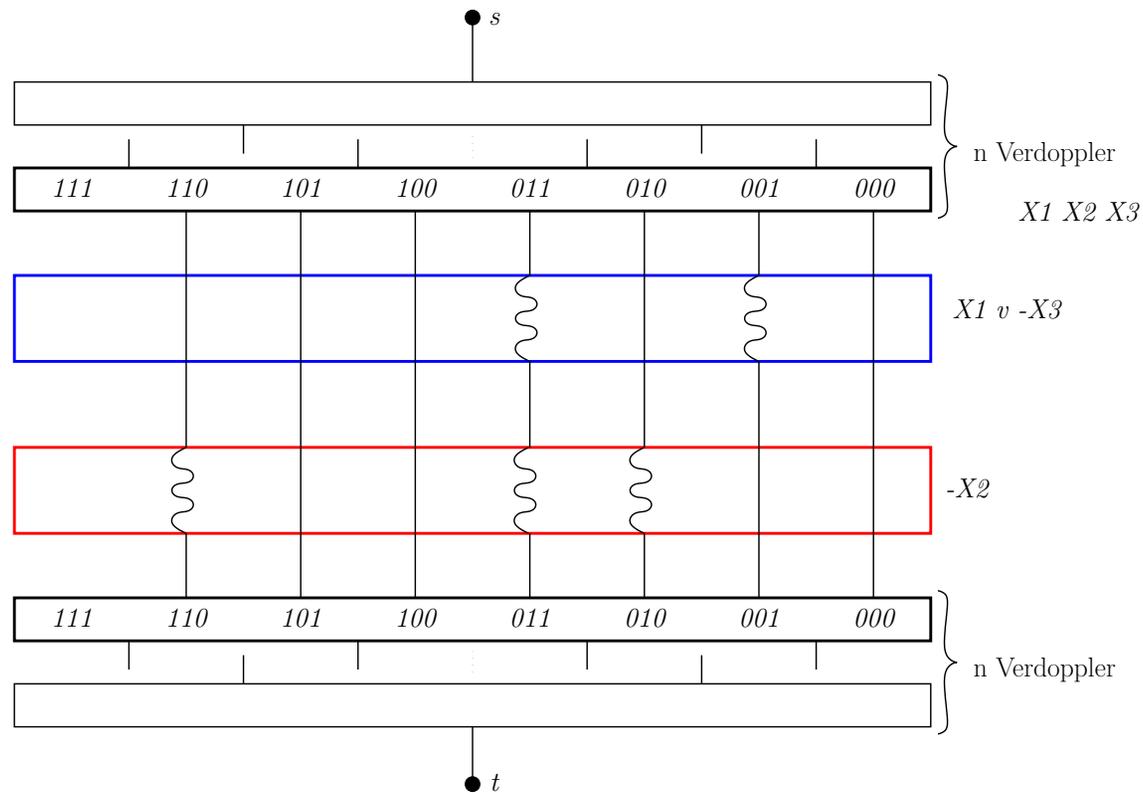
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



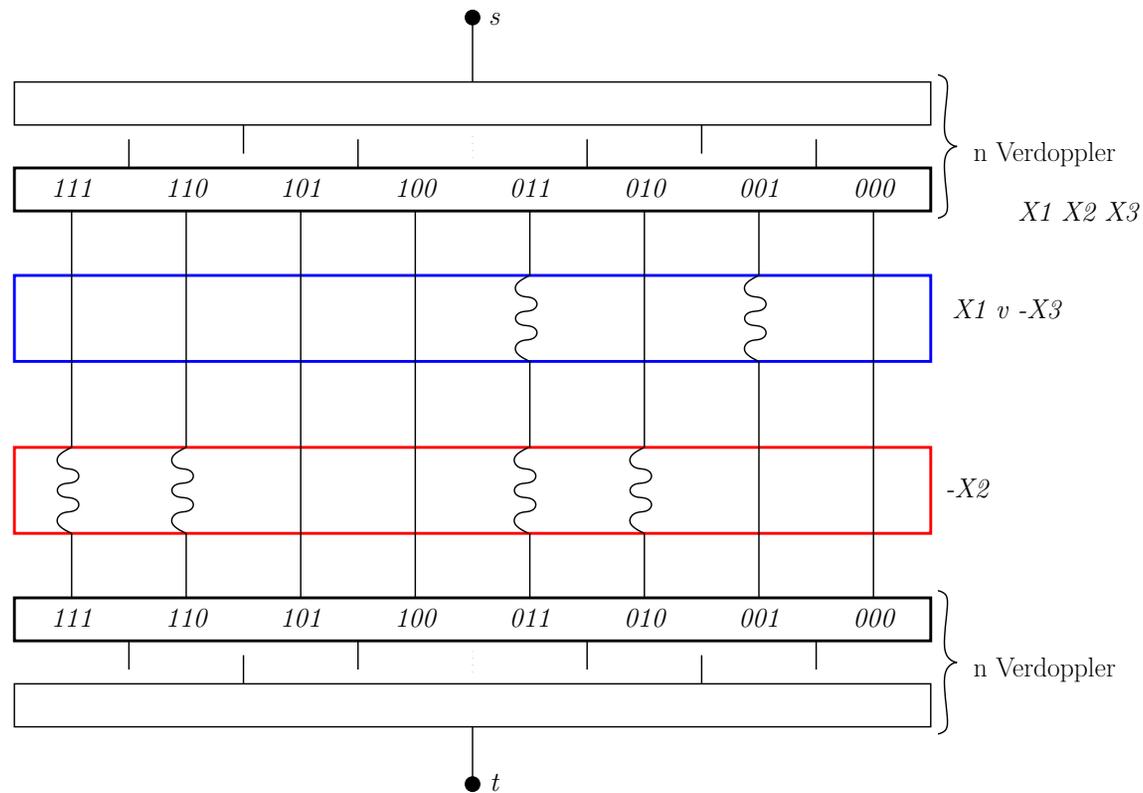
Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



Parcours erzeugen: Prinzip

Beispiel: $(X1 \vee \neg X3) \wedge (\neg X2)$



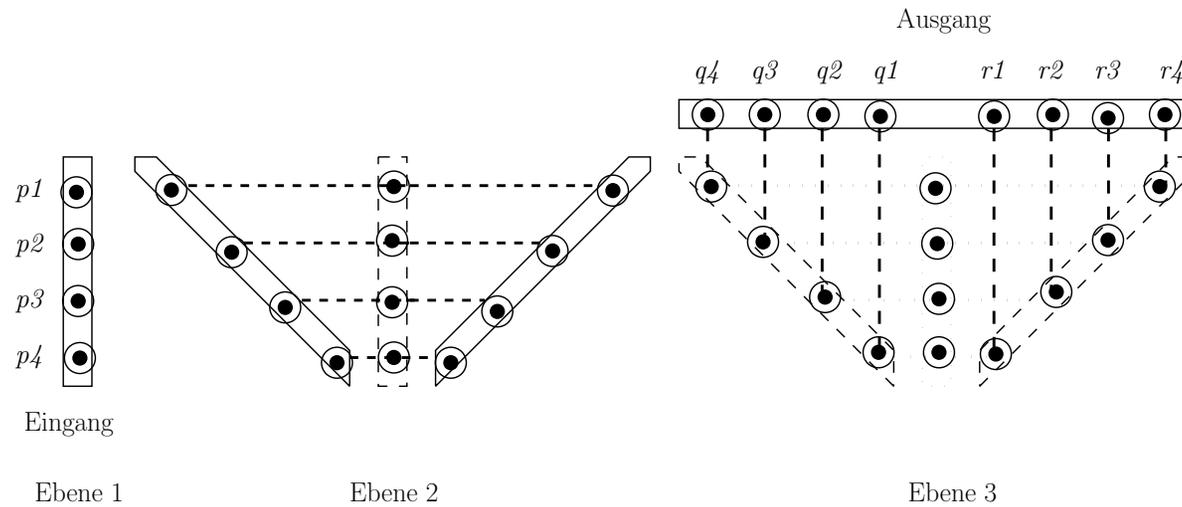
Komponenten: Verdoppler!

Komponenten: Verdoppler!

Dünne Platten mit Schlitzen eng hintereinander!

Komponenten: Verdoppler!

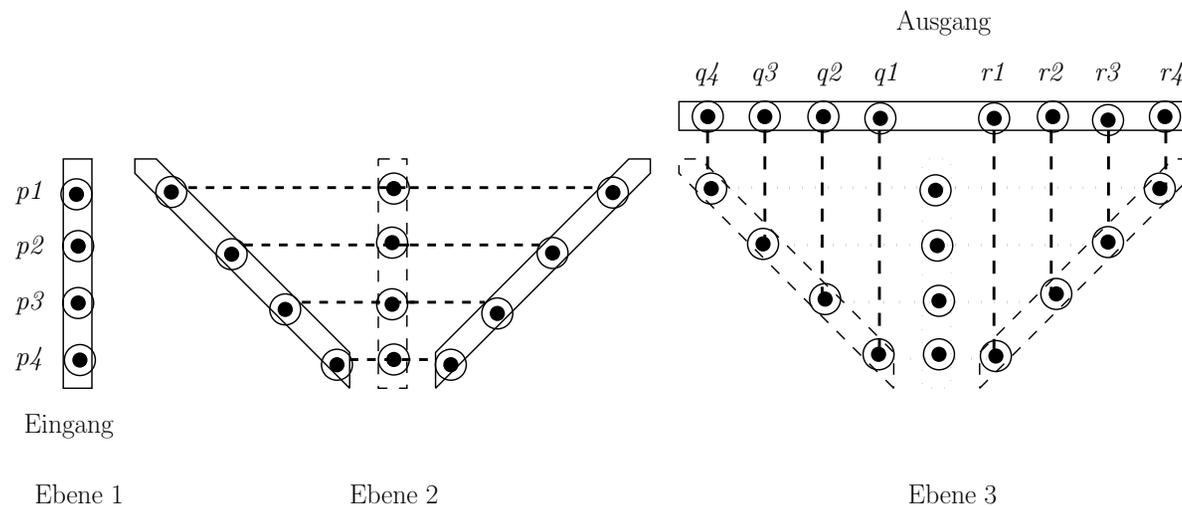
Dünne Platten mit Schlitzern eng hintereinander!



Sukzessive 2^n ungefähr gleichlange Wege erzeugen!

Komponenten: Verdoppler!

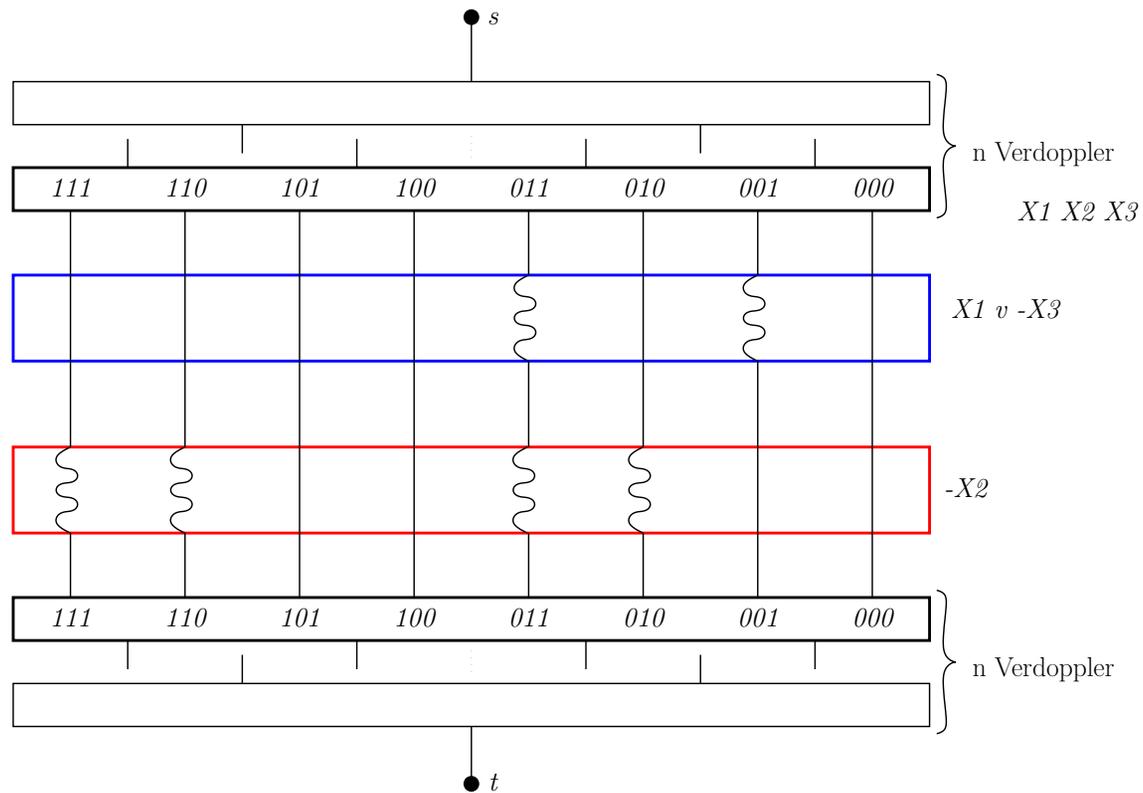
Dünne Platten mit Schlitzern eng hintereinander!



Sukzessive 2^n ungefähr gleichlange Wege erzeugen!

Kantenreihenfolge ist gleich!

Gesamtprinzip: Nacheinander Klauseln!



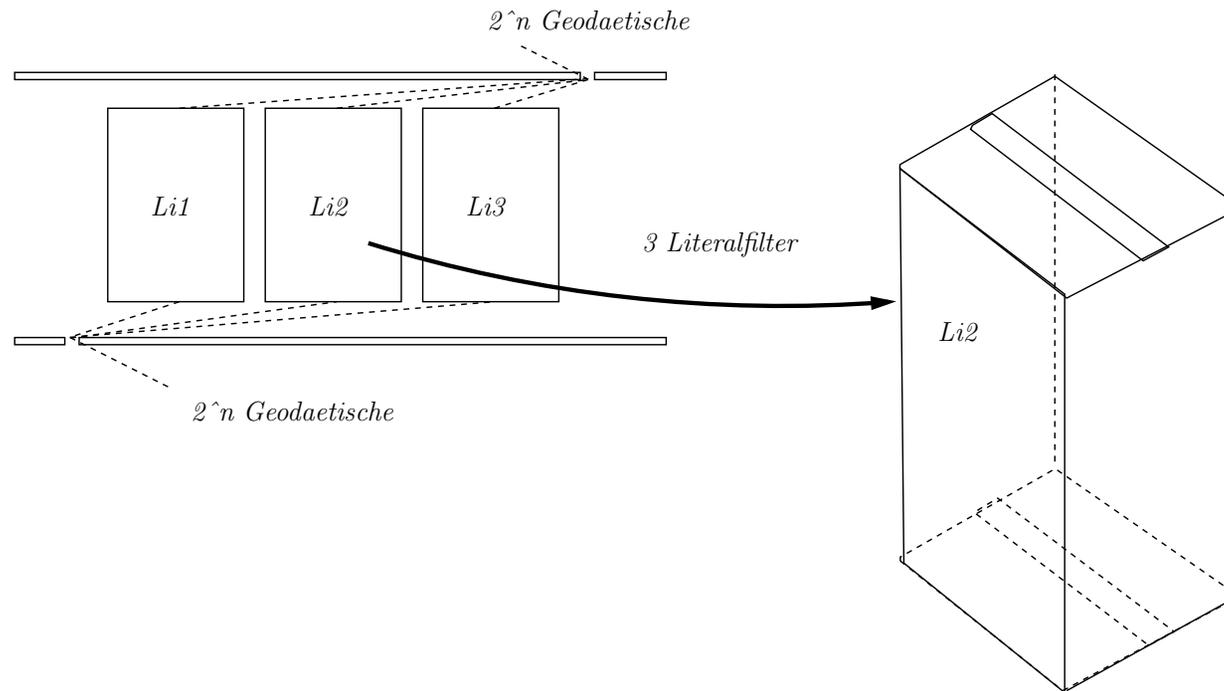
Komponenten: Klauselfilter

Komponenten: Klauselfilter

Sukzessive durch die Klauseln schicken! Auf Literale aufteilen!
Dünne Platten!

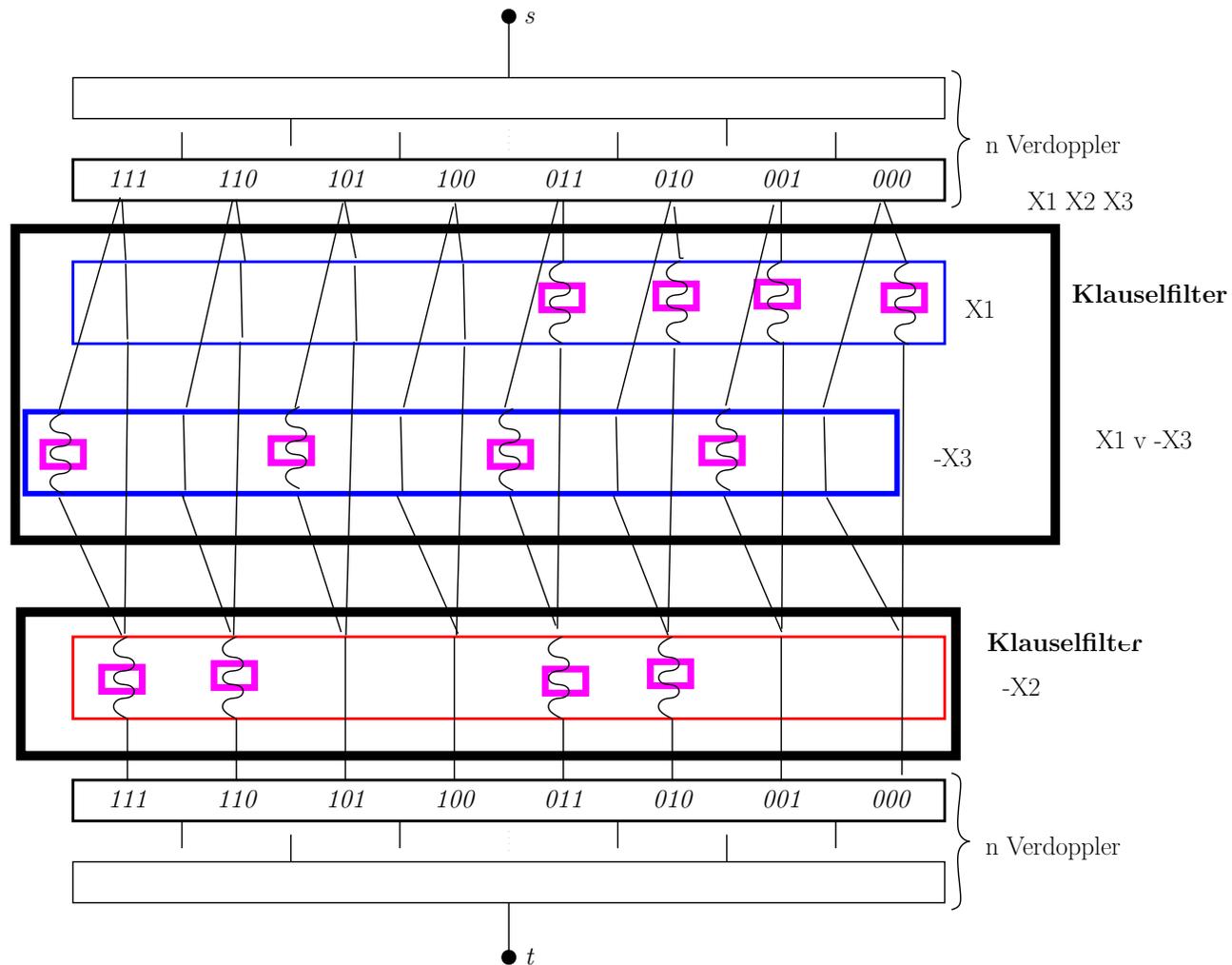
Komponenten: Klauselfilter

Sukzessive durch die Klauseln schicken! Auf Literale aufteilen!
Dünne Platten!



Gleich lang, bis auf das, was in den Literalfiltern passiert!

Gesamtprinzip: Einzelne Literale



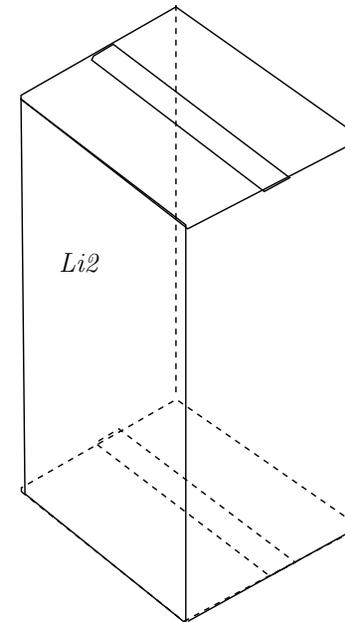
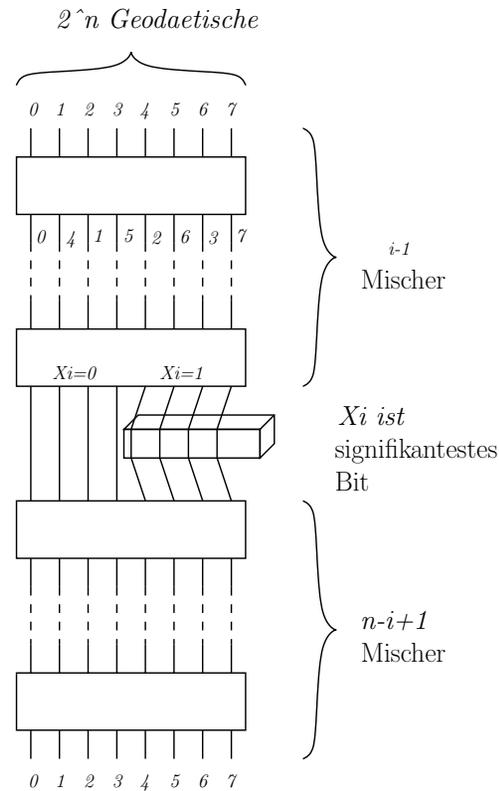
Komponenten: Literalfilter

Komponenten: Literalfilter

Wege für signifikantes Bit sammeln!

Komponenten: Literalfilter

Wege für signifikantes Bit sammeln!



Falls X_i dann $X_i = 0$ verlängern! Falls $\neg X_i$ dann $X_i = 1$ verlängern!

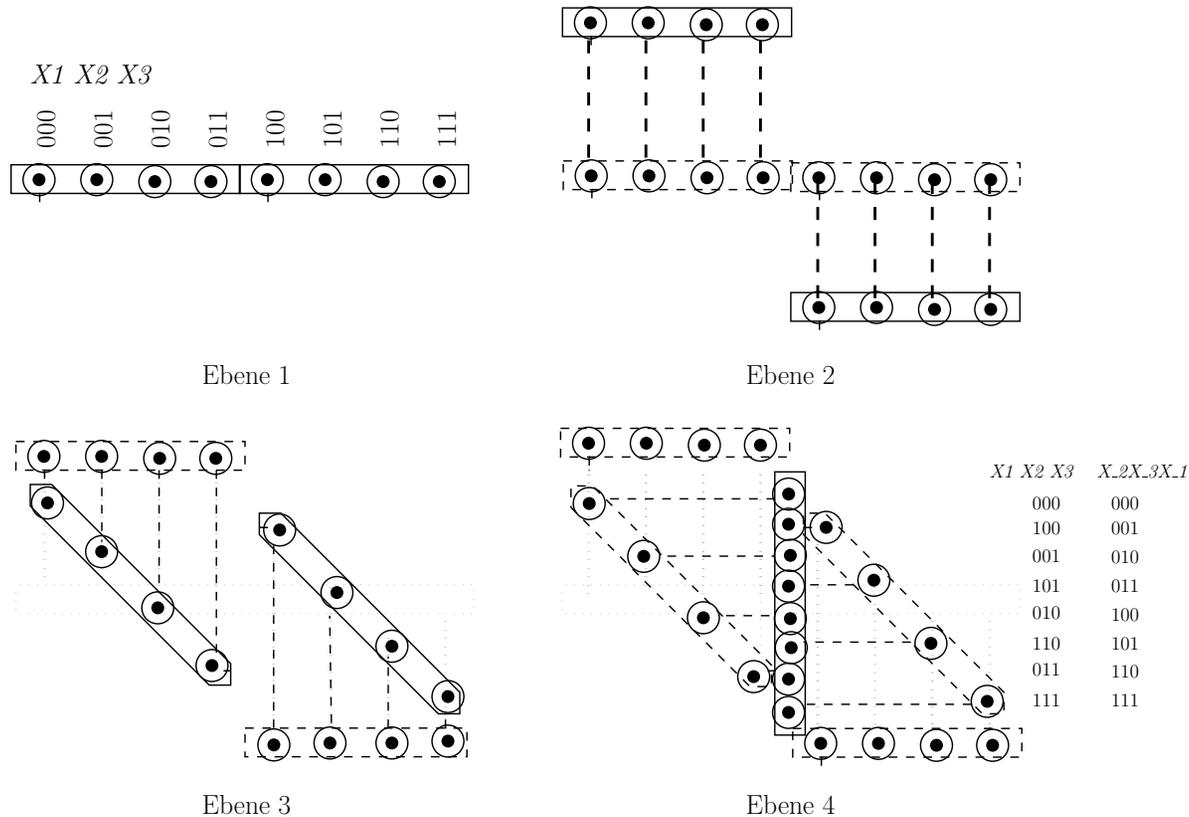
n **Mischer pro Literalfilter!!**

n **Mischer pro Literalfilter!!**

Ein Mischer erzeugt Bitverschiebung der Wege um 1! Alle bleiben gleich lang!!

n Mischer pro Literalfilter!!

Ein Mischer erzeugt Bitverschiebung der Wege um 1! Alle bleiben gleich lang!!



Kürzeste Wege Alg. für P_α

Kürzeste Wege Alg. für P_α

Ein Weg, der nicht verlängert wird entspricht genau einer Belegung, die die Formel erfüllt!!!

Kürzeste Wege Alg. für P_α

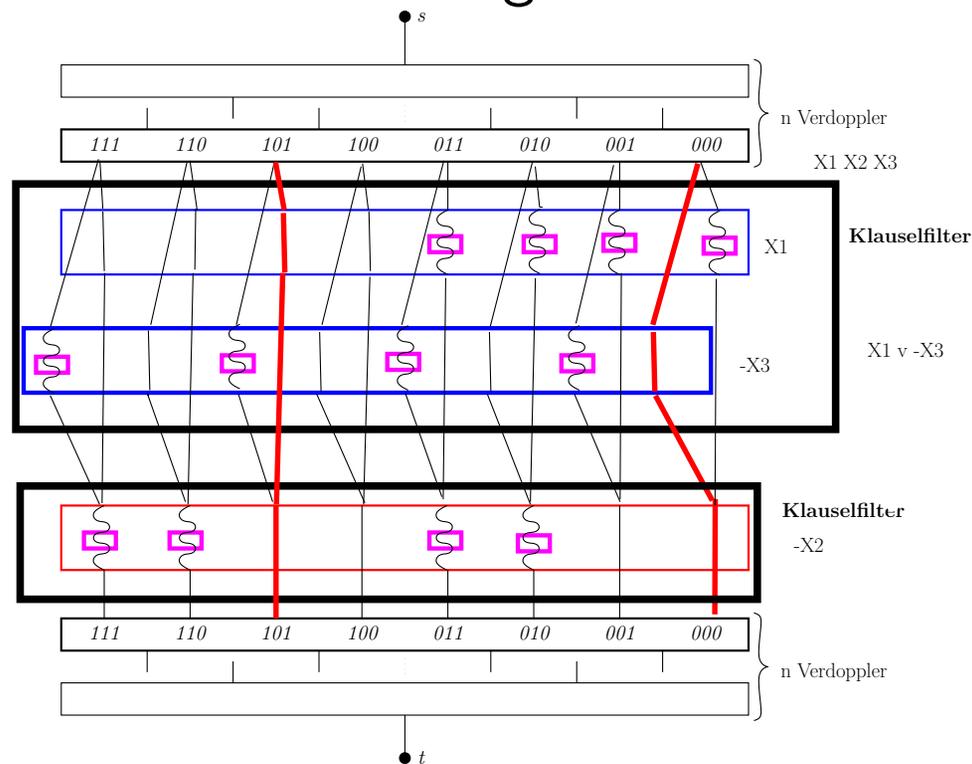
Ein Weg, der nicht verlängert wird entspricht genau einer Belegung, die die Formel erfüllt!!!

Das kann man der Kantenreihenfolge entnehmen!!

Kürzeste Wege Alg. für P_α

Ein Weg, der nicht verlängert wird entspricht genau einer Belegung, die die Formel erfüllt!!!

Das kann man der Kantenreihenfolge entnehmen!!



Konstruktion insgesamt!!

Konstruktion insgesamt!!

- $2n$ Verdoppler: $O(n)$ Kanten

Konstruktion insgesamt!!

- $2n$ Verdoppler: $O(n)$ Kanten
- m Klauselfilter: je Klauselfilter
 - 3 Literalfilter

Konstruktion insgesamt!!

- $2n$ Verdoppler: $O(n)$ Kanten
- m Klauselfilter: je Klauselfilter
 - 3 Literalfilter
 - n Mischer je Filter

Konstruktion insgesamt!!

- $2n$ Verdoppler: $O(n)$ Kanten
- m Klauselfilter: je Klauselfilter
 - 3 Literalfilter
 - n Mischer je Filter
- Insgesamt $O(mn)$ Kanten

Konstruktion insgesamt!!

- $2n$ Verdoppler: $O(n)$ Kanten
- m Klauselfilter: je Klauselfilter
 - 3 Literalfilter
 - n Mischer je Filter
- Insgesamt $O(mn)$ Kanten
- In polynomieller Zeit konstruierbar

Ergebnis!!!

Ergebnis!!!

Theorem 1.38 (Canny/Reif): Bestimmung der optimalen Kantenfolge bei der Berechnung Kürzester Wege in polyedrischer Szene in 3D ist NP hart.

Ergebnis!!!

Theorem 1.38 (Canny/Reif): Bestimmung der optimalen Kantenfolge bei der Berechnung Kürzester Wege in polyedrischer Szene in 3D ist NP hart.

Beweis!!