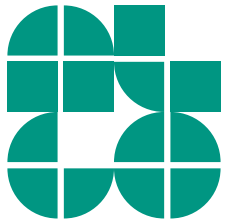
Vorlesung Algorithmische Kartografie

Trajektorienbasierte Beschriftung

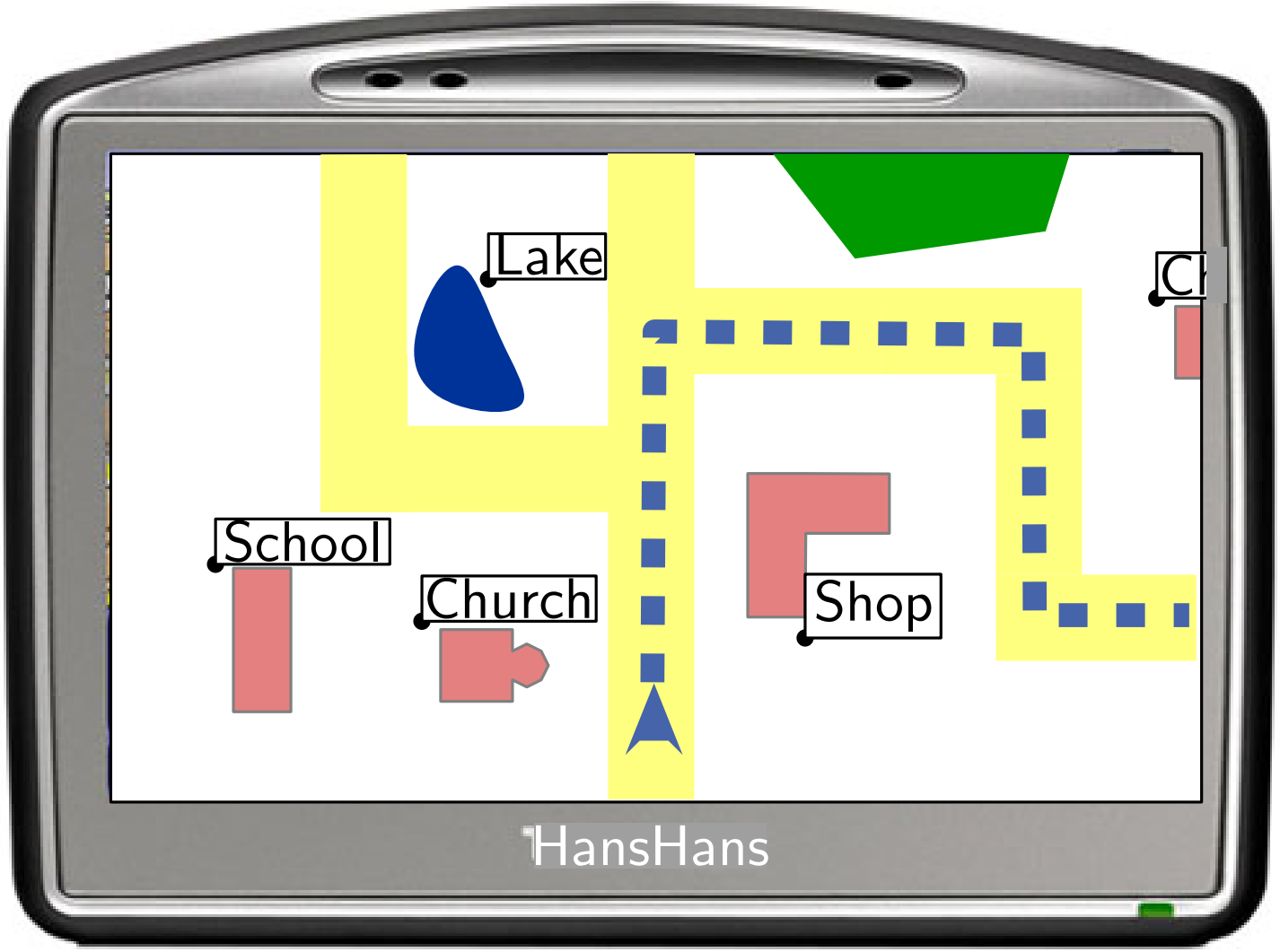
LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Benjamin Niedermann

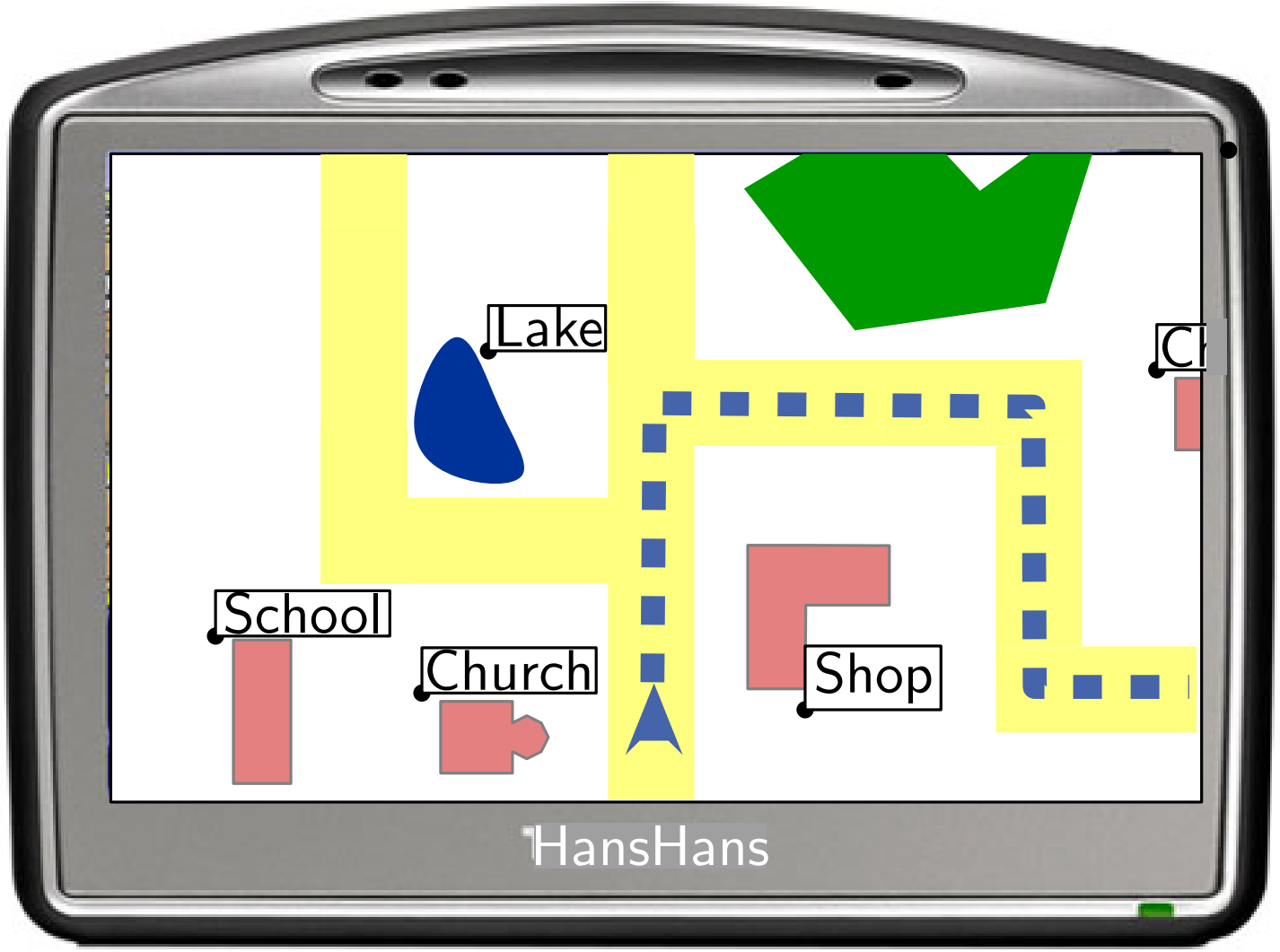
11.06.2015



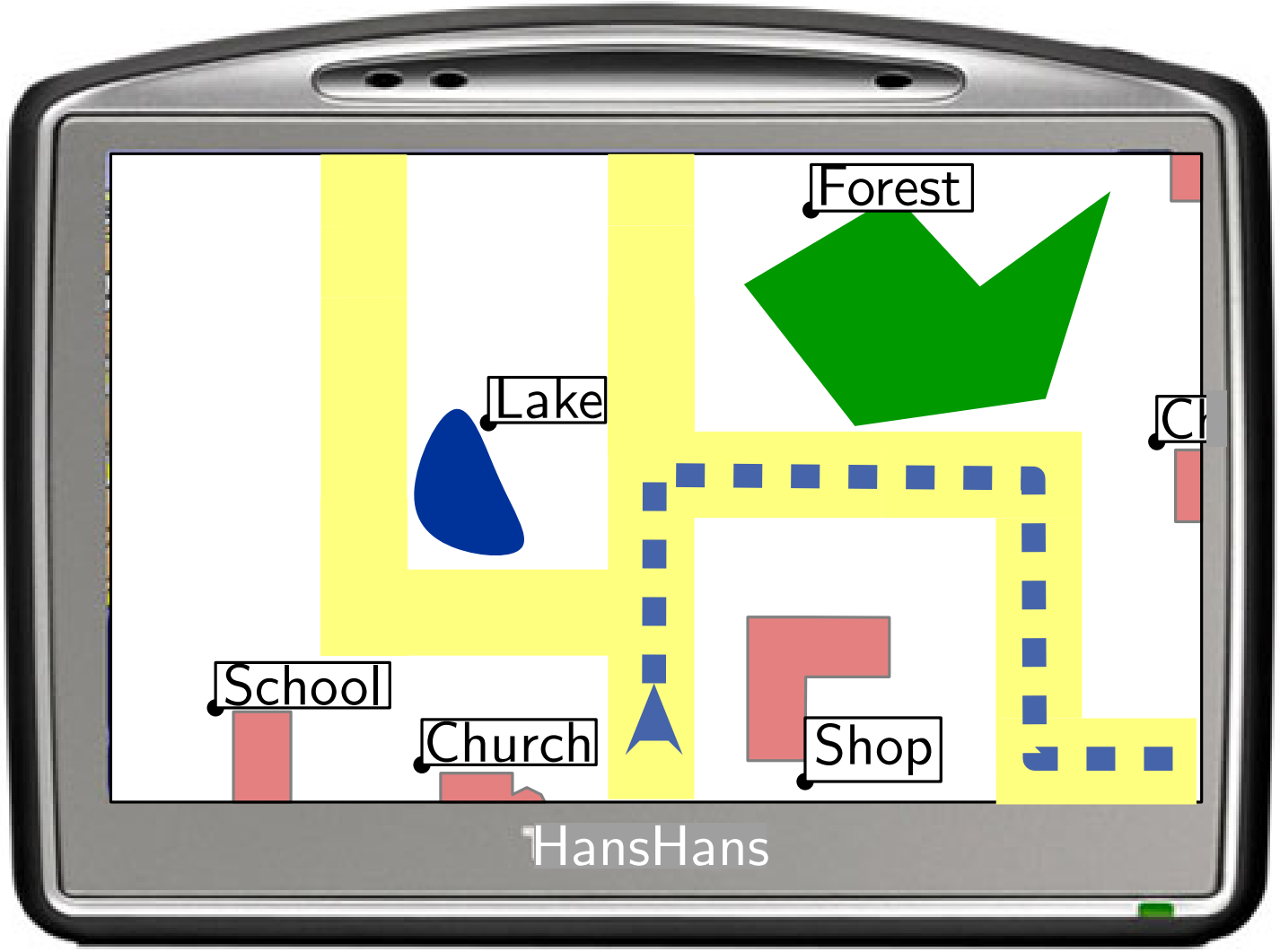
Motivation



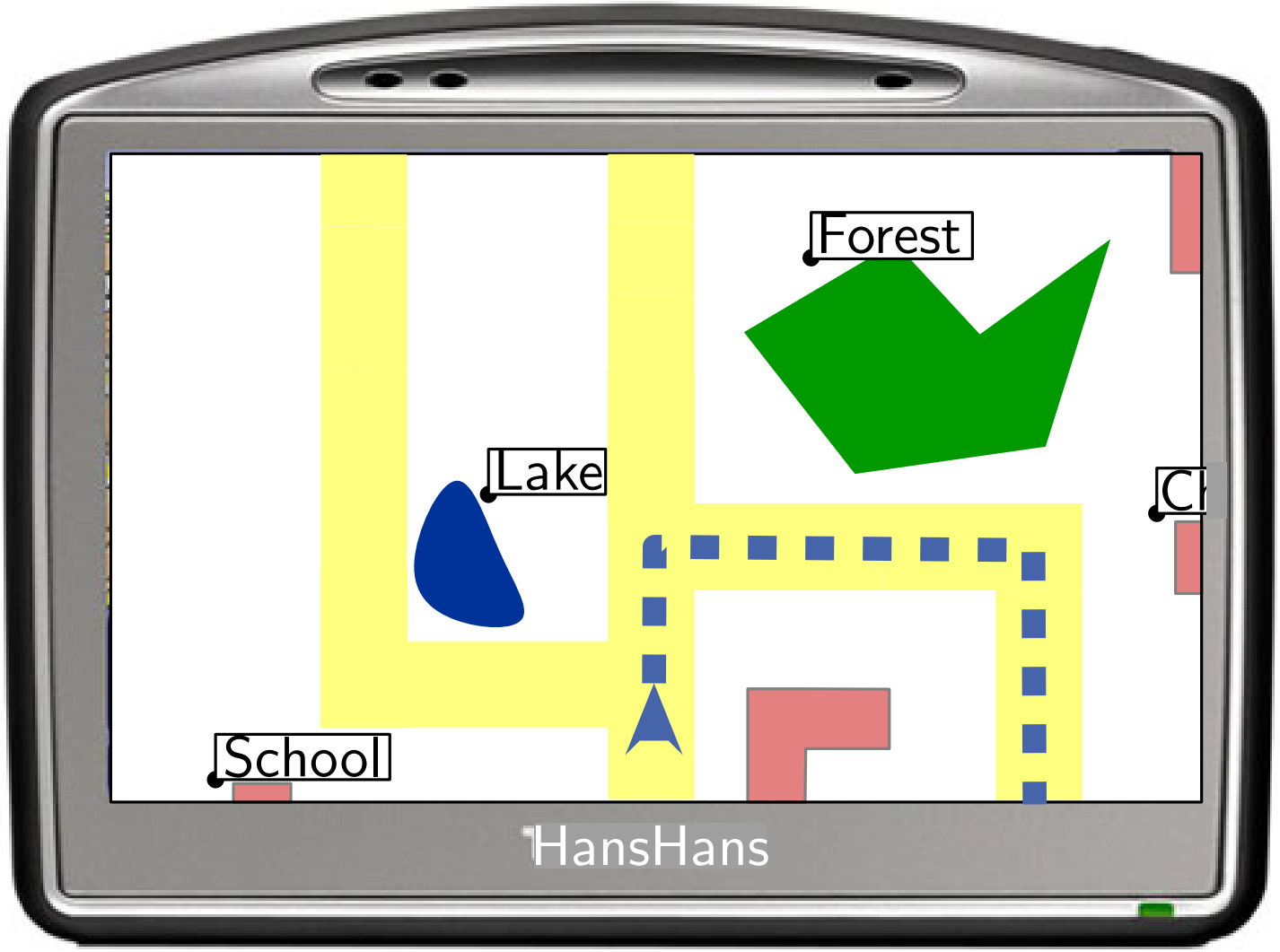
Motivation



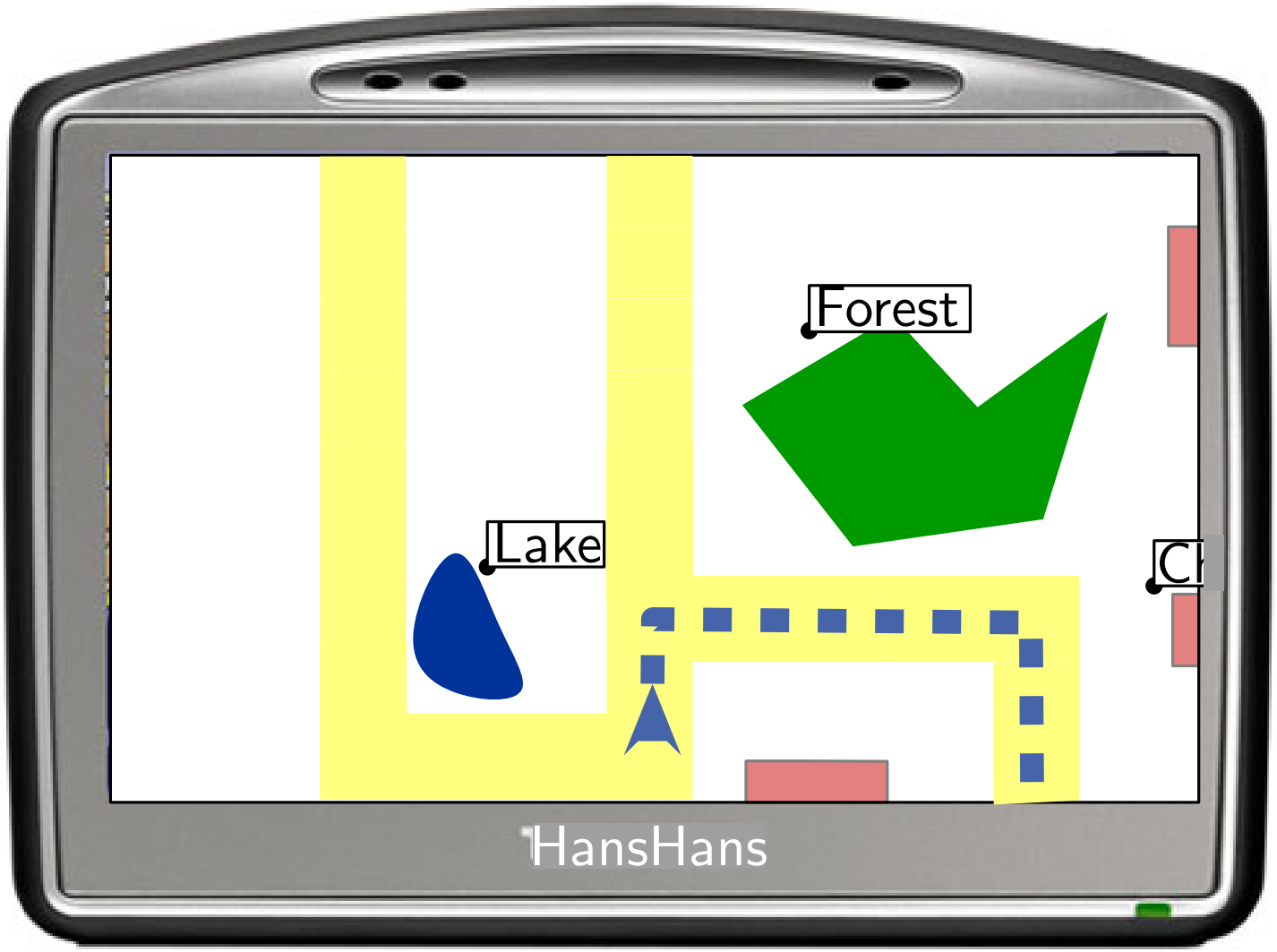
Motivation



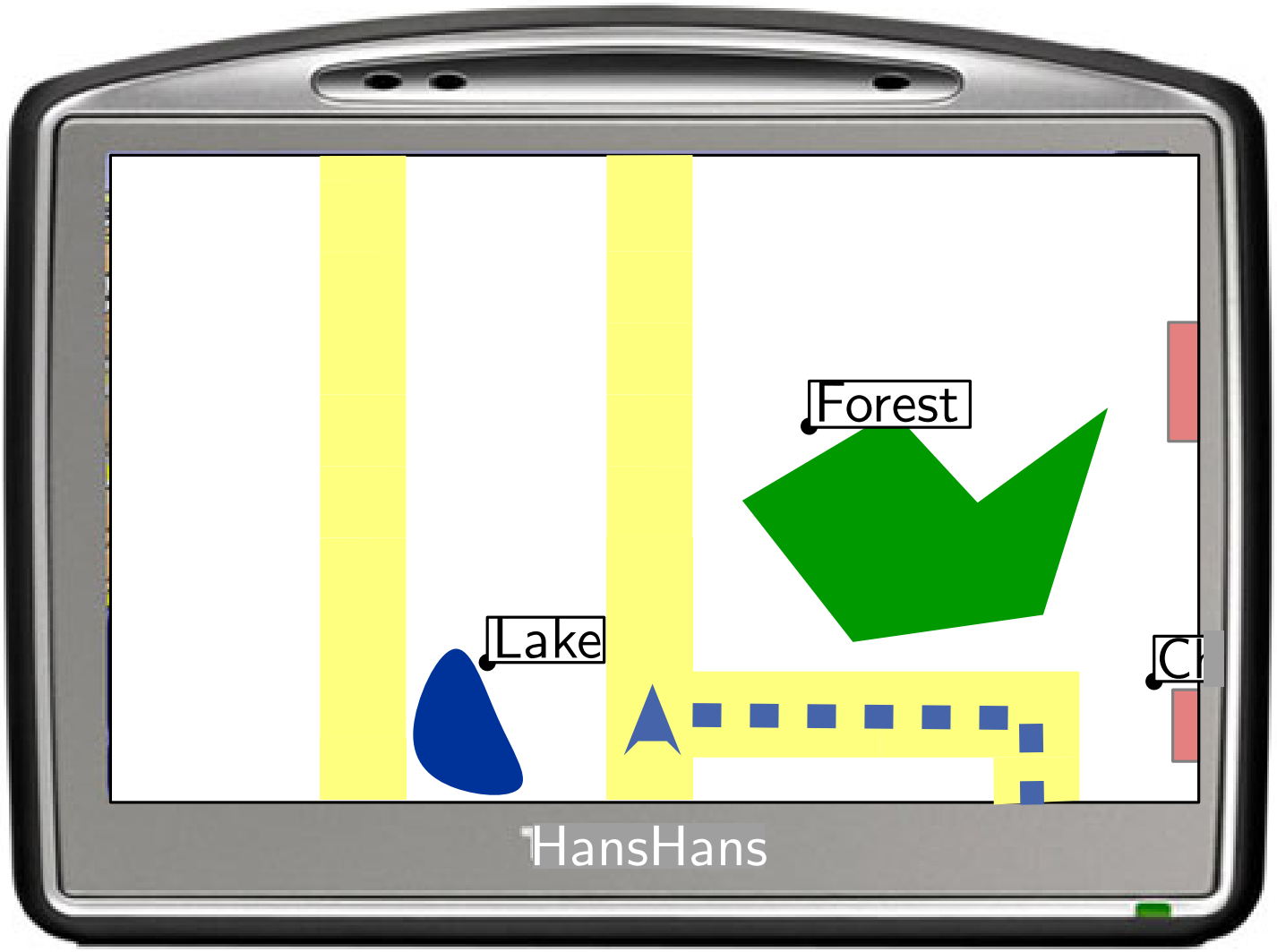
Motivation



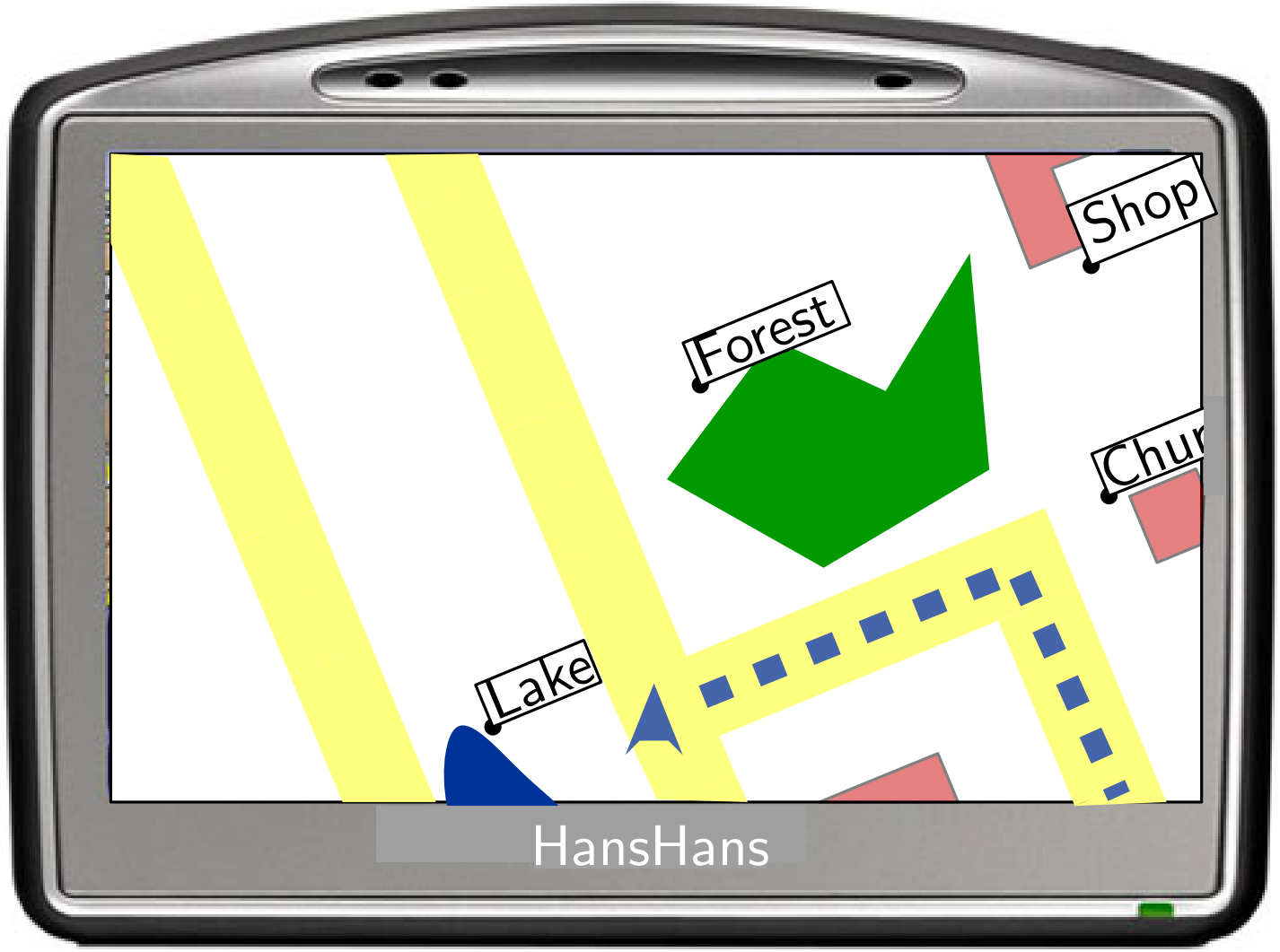
Motivation



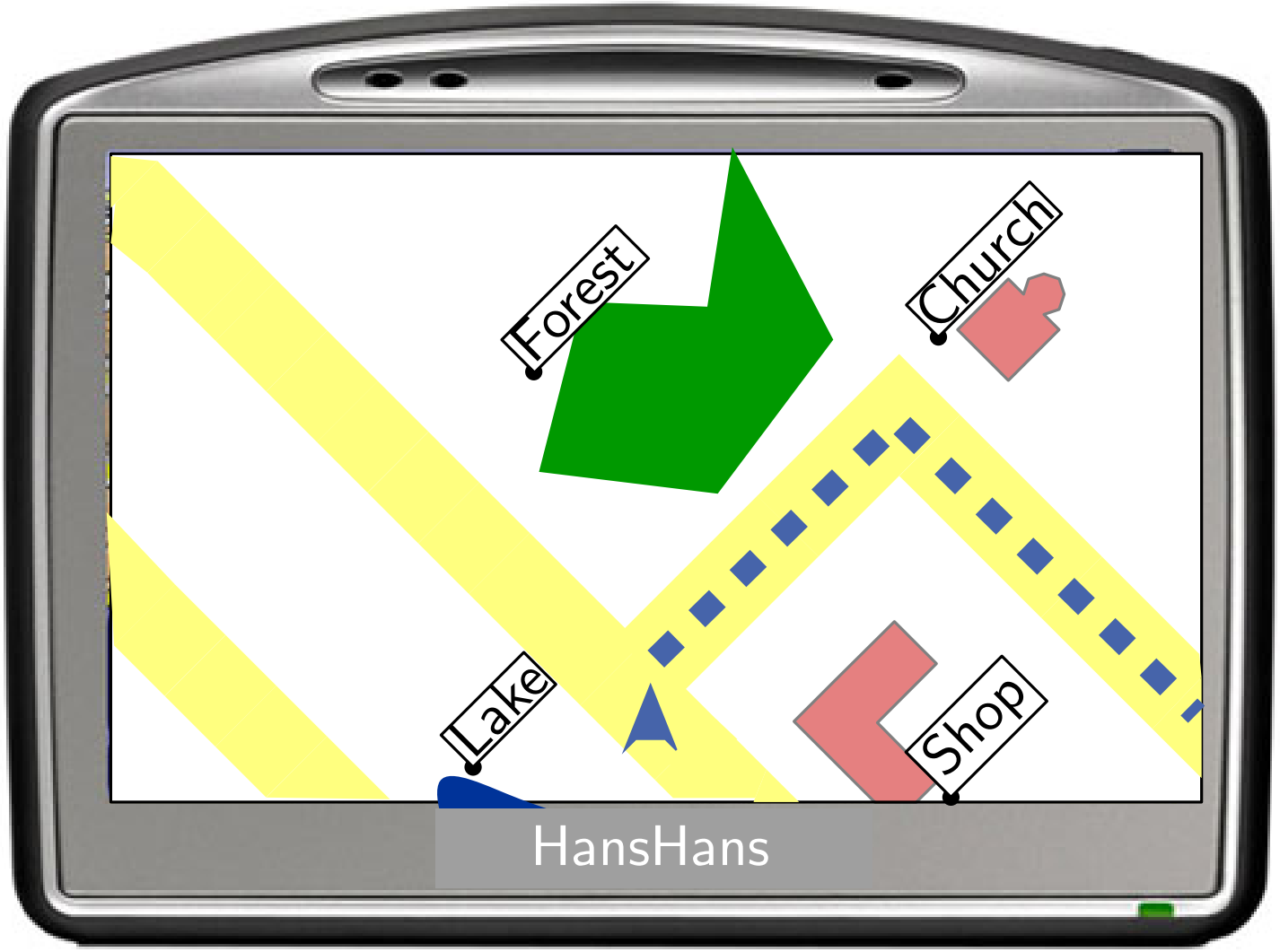
Motivation



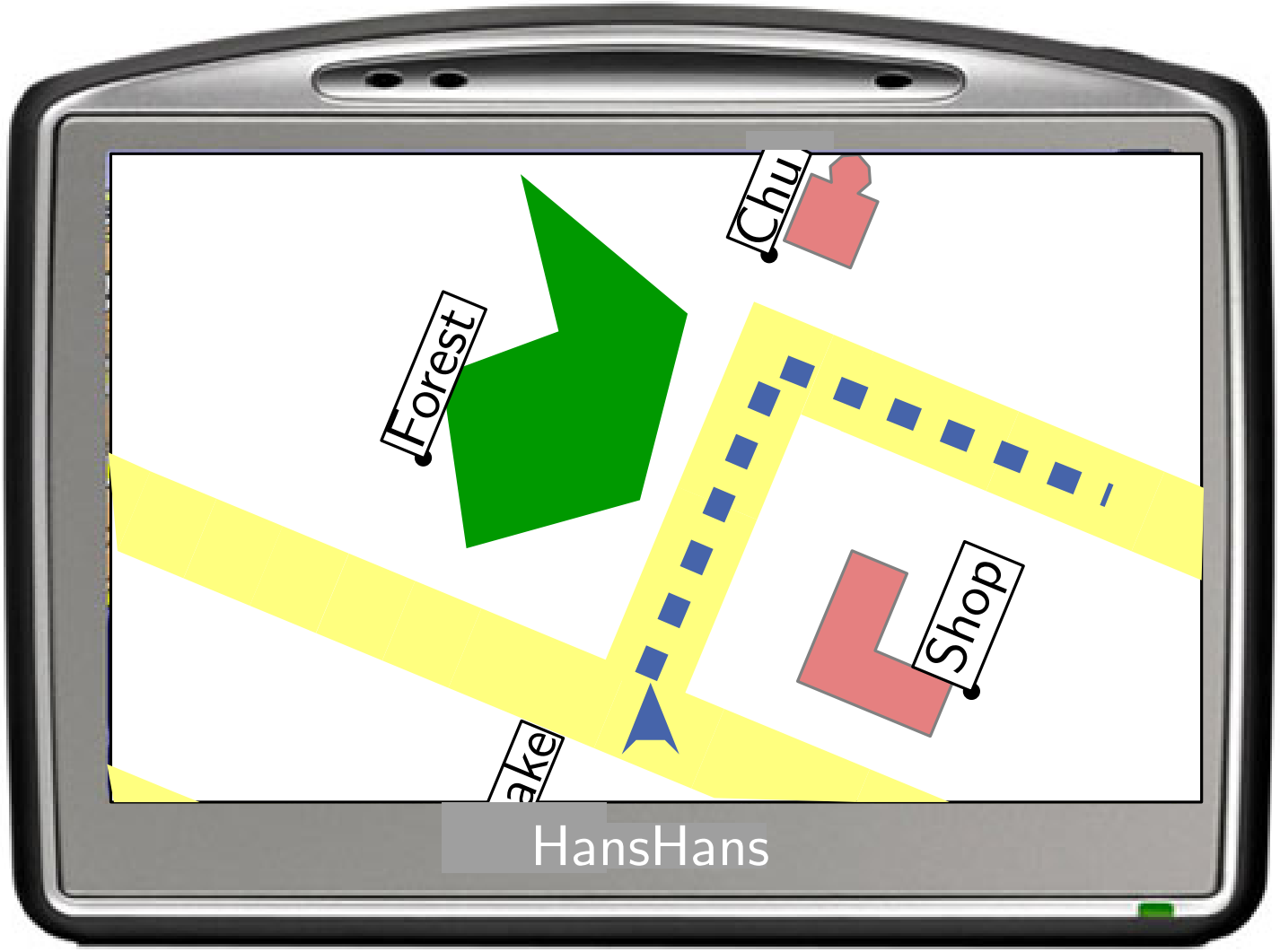
Motivation



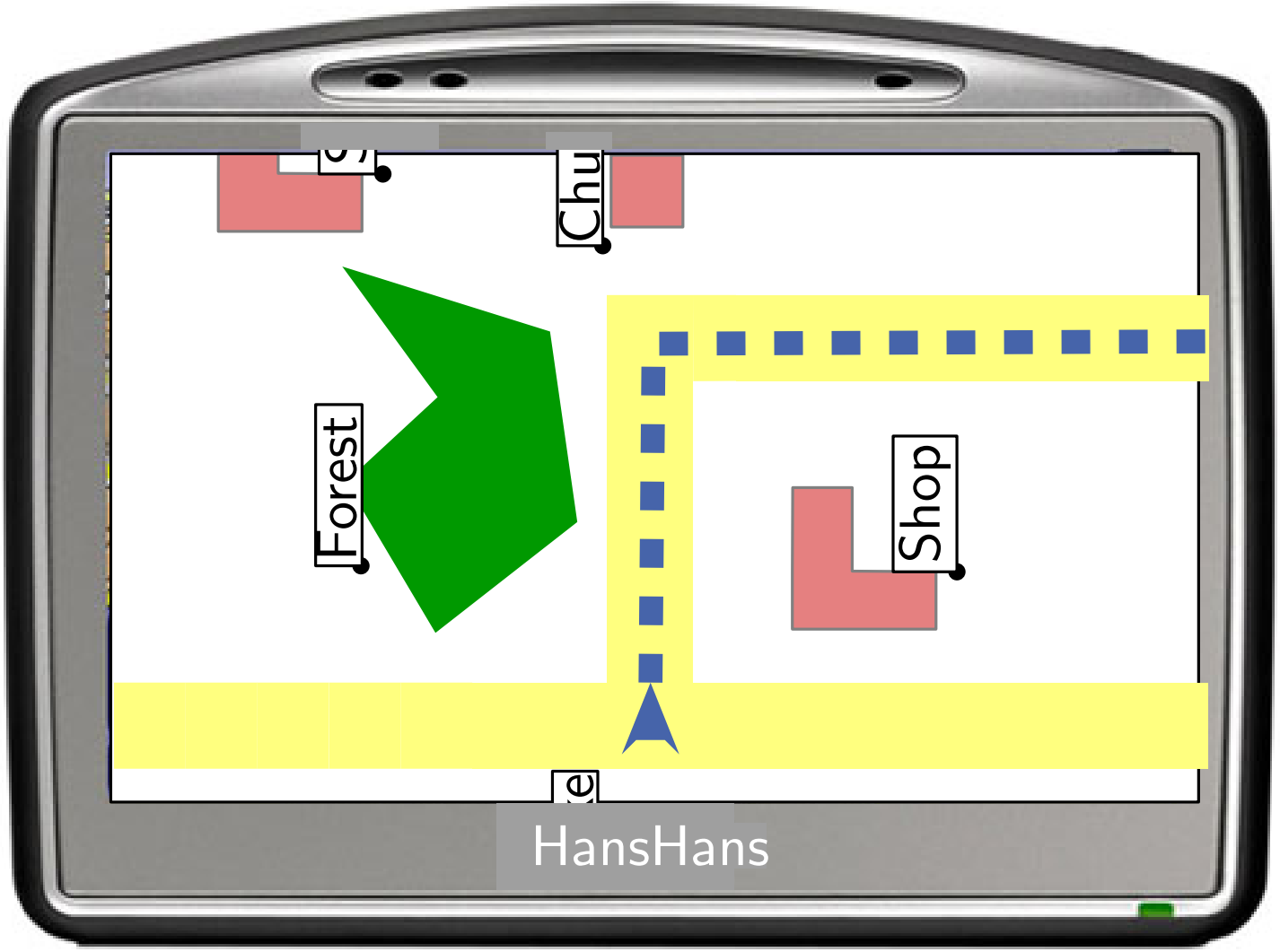
Motivation



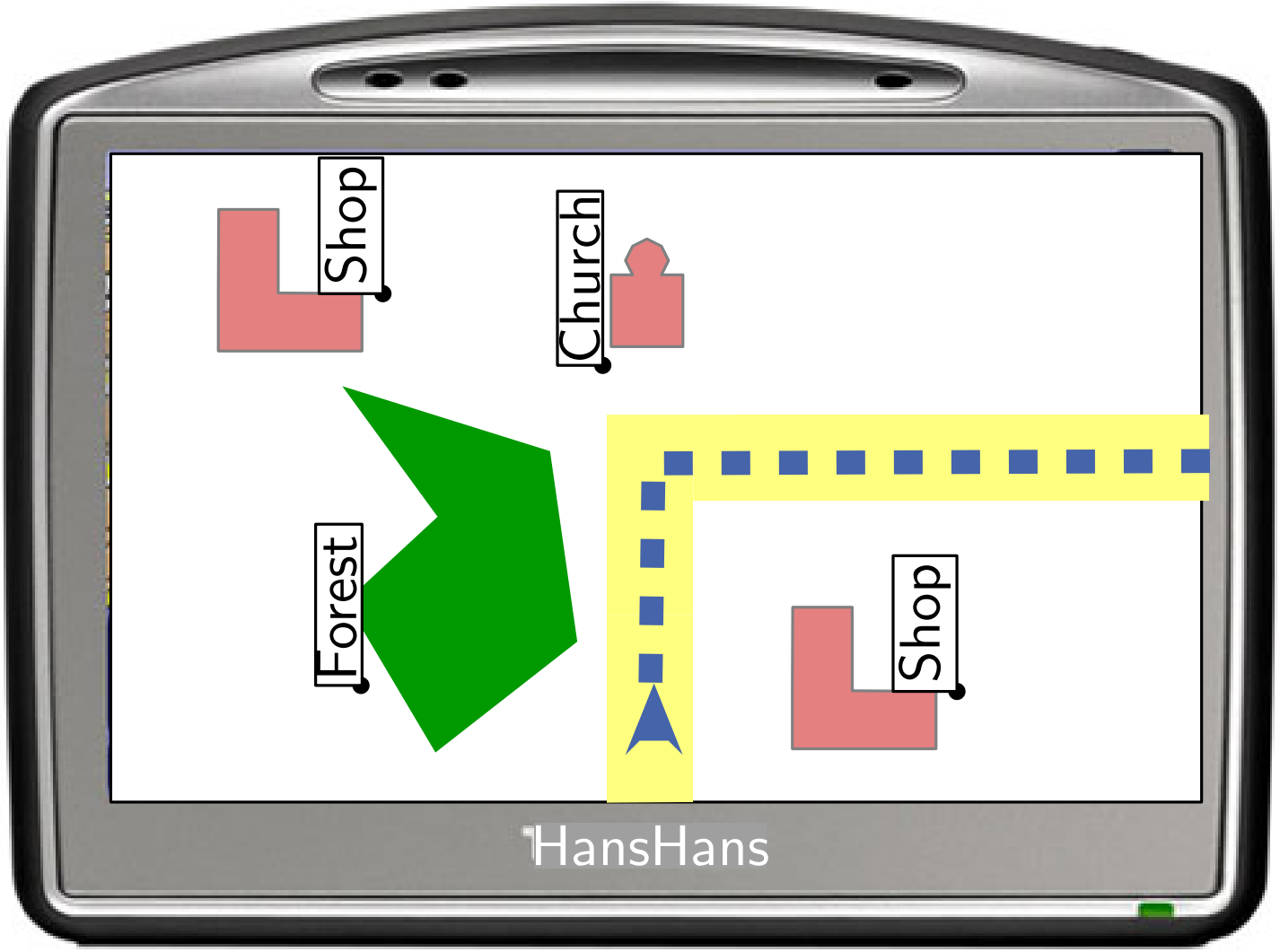
Motivation



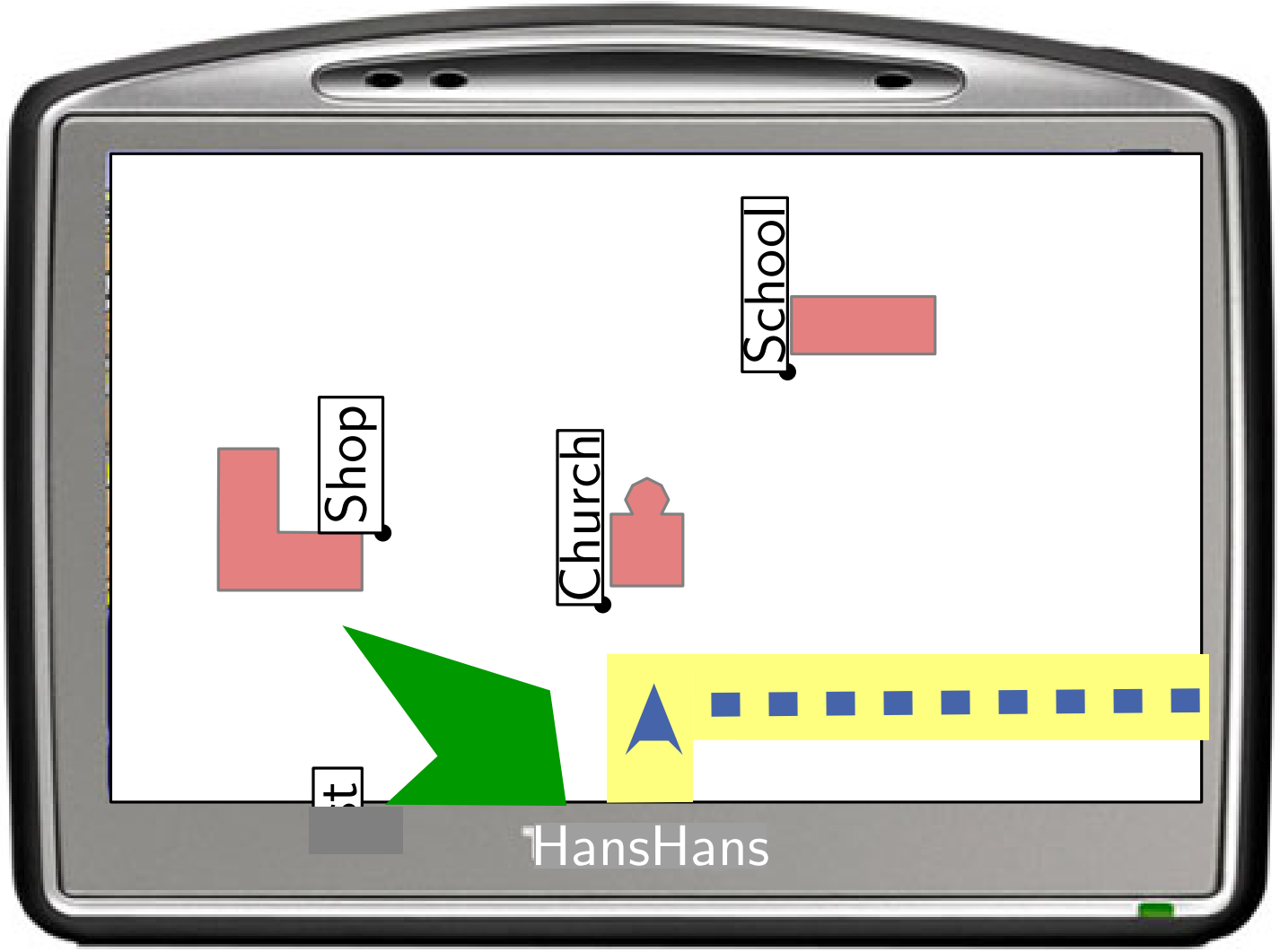
Motivation



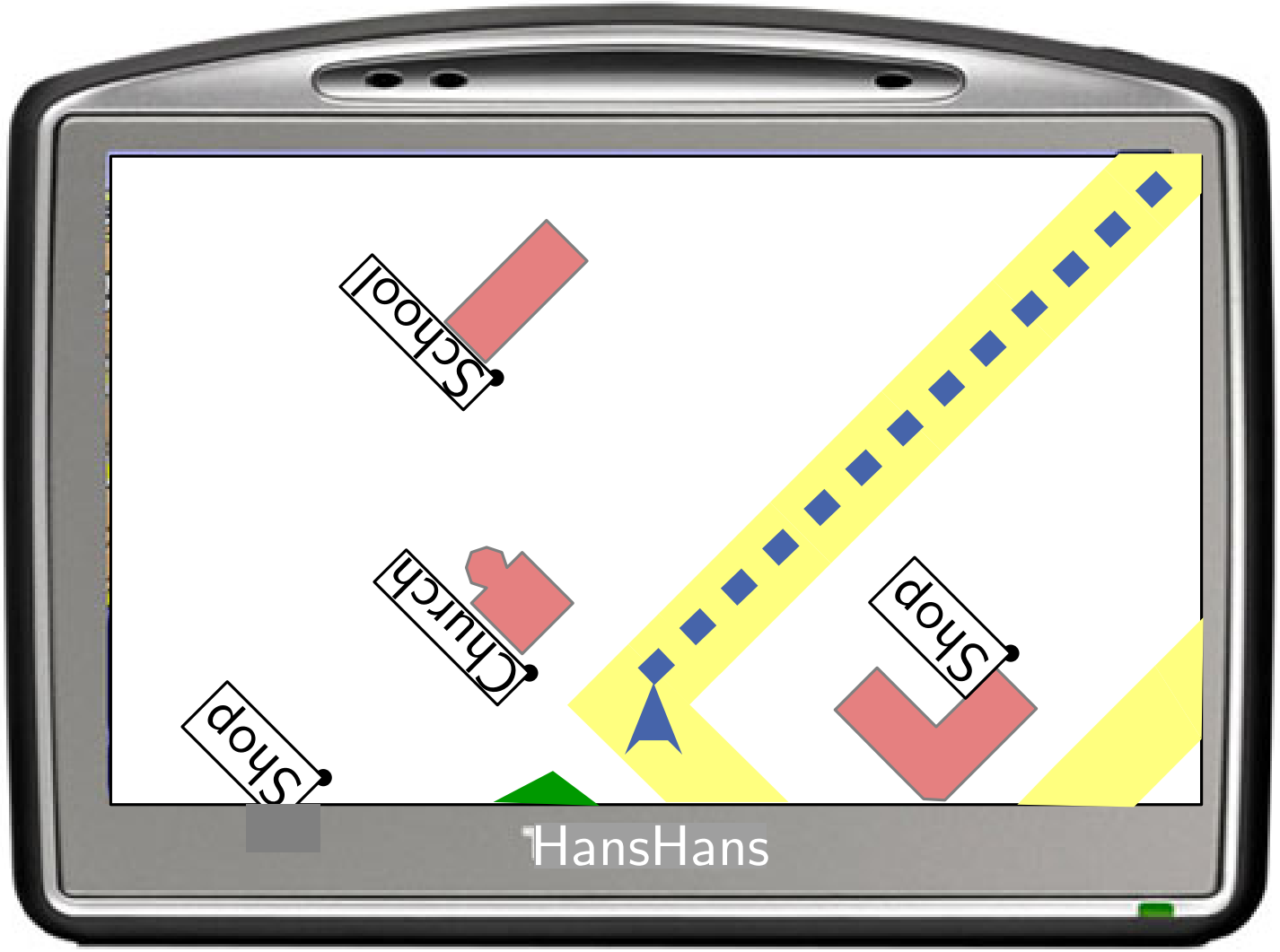
Motivation



Motivation

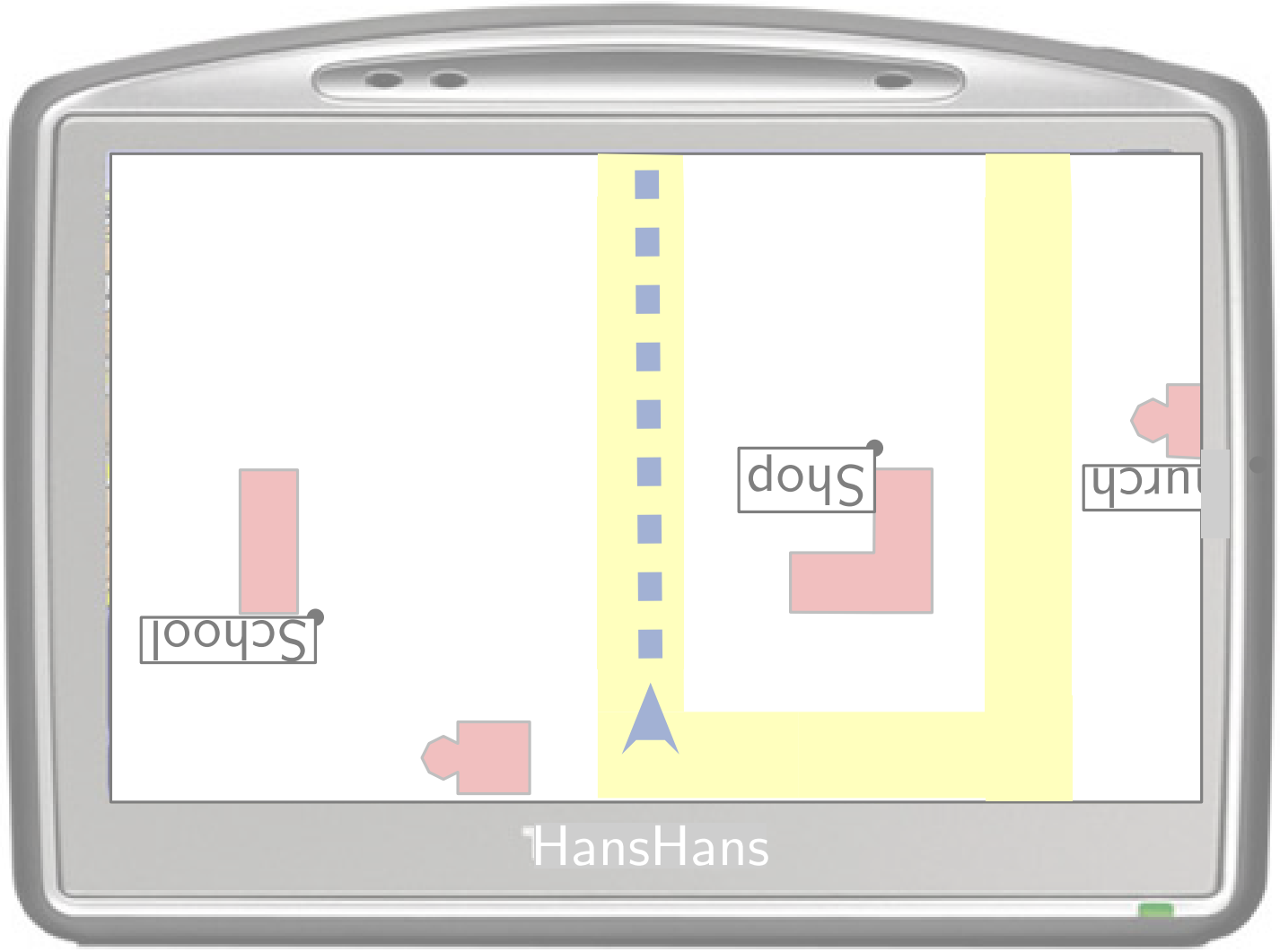


Motivation

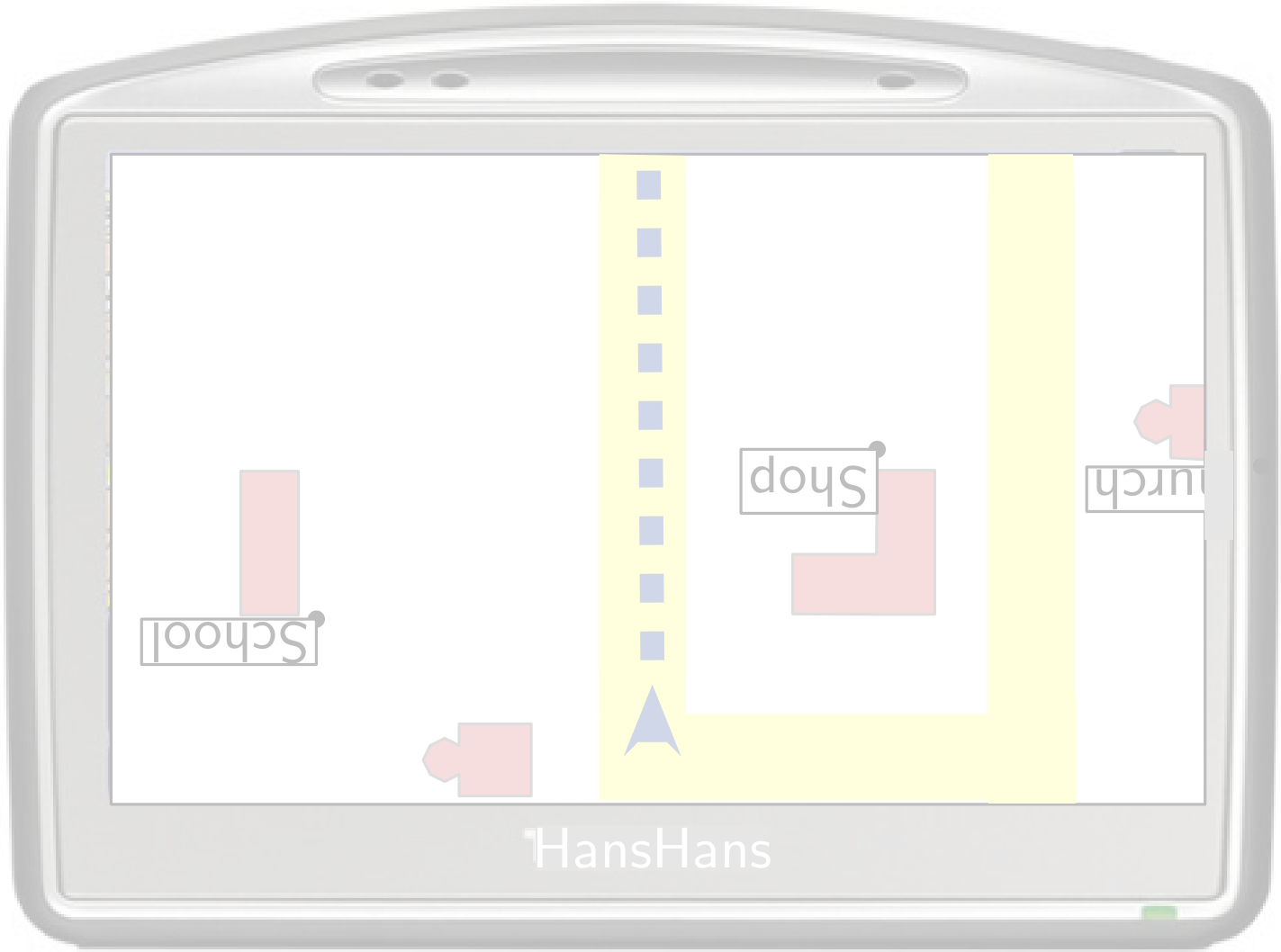


Motivation

Motivation



Motivation



Motivation

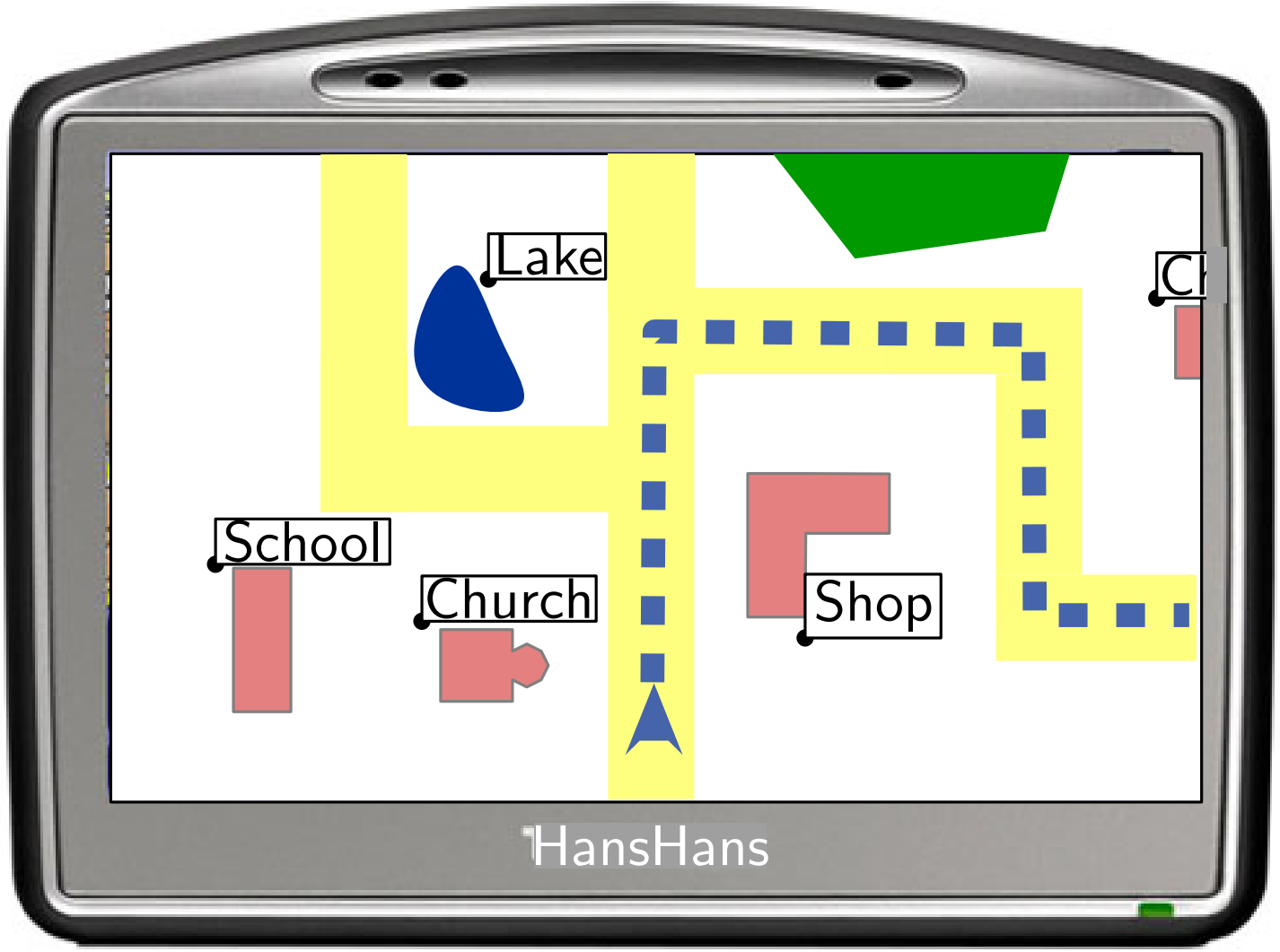
Motivation

Motivation

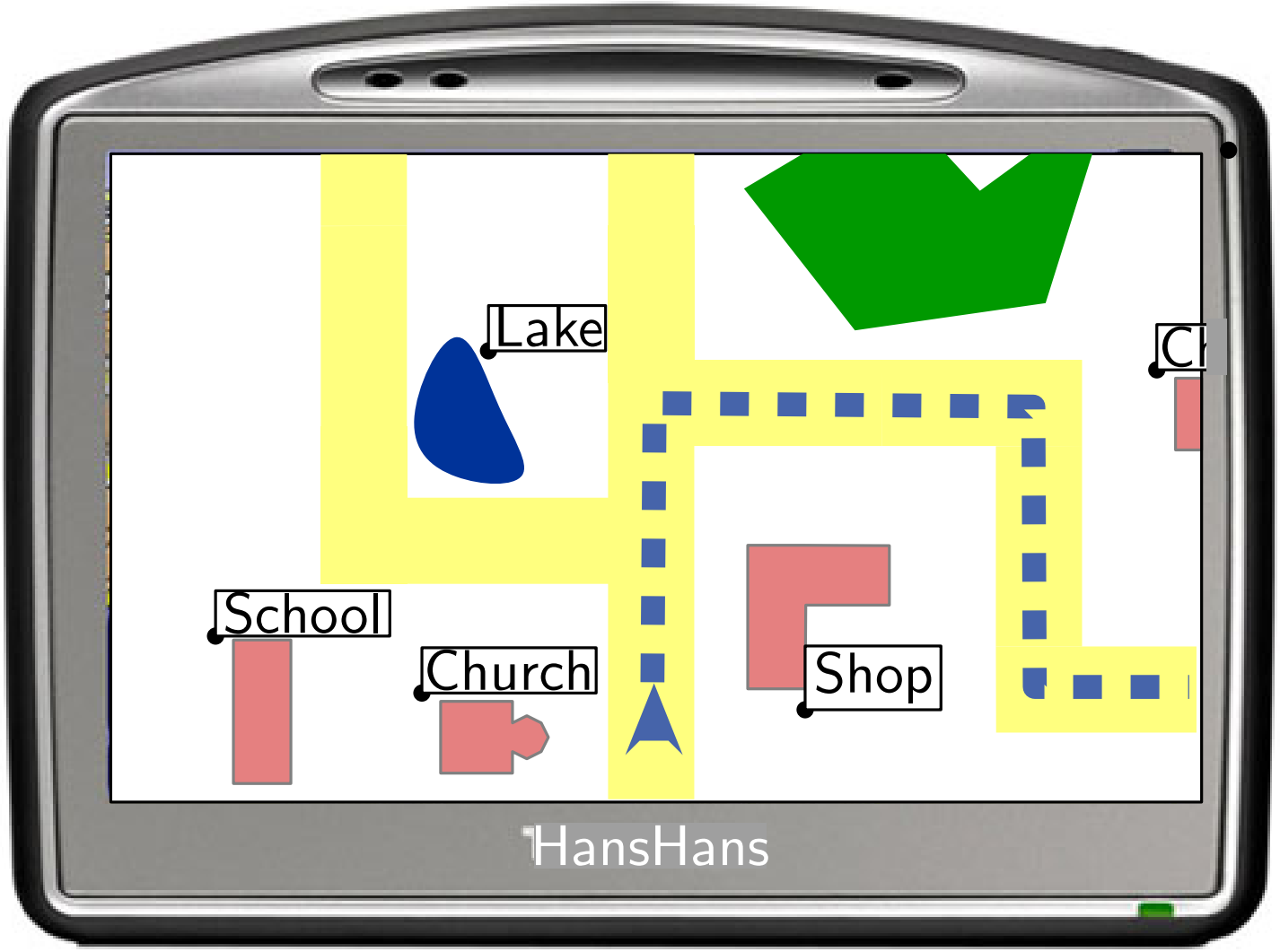
Motivation

Motivation

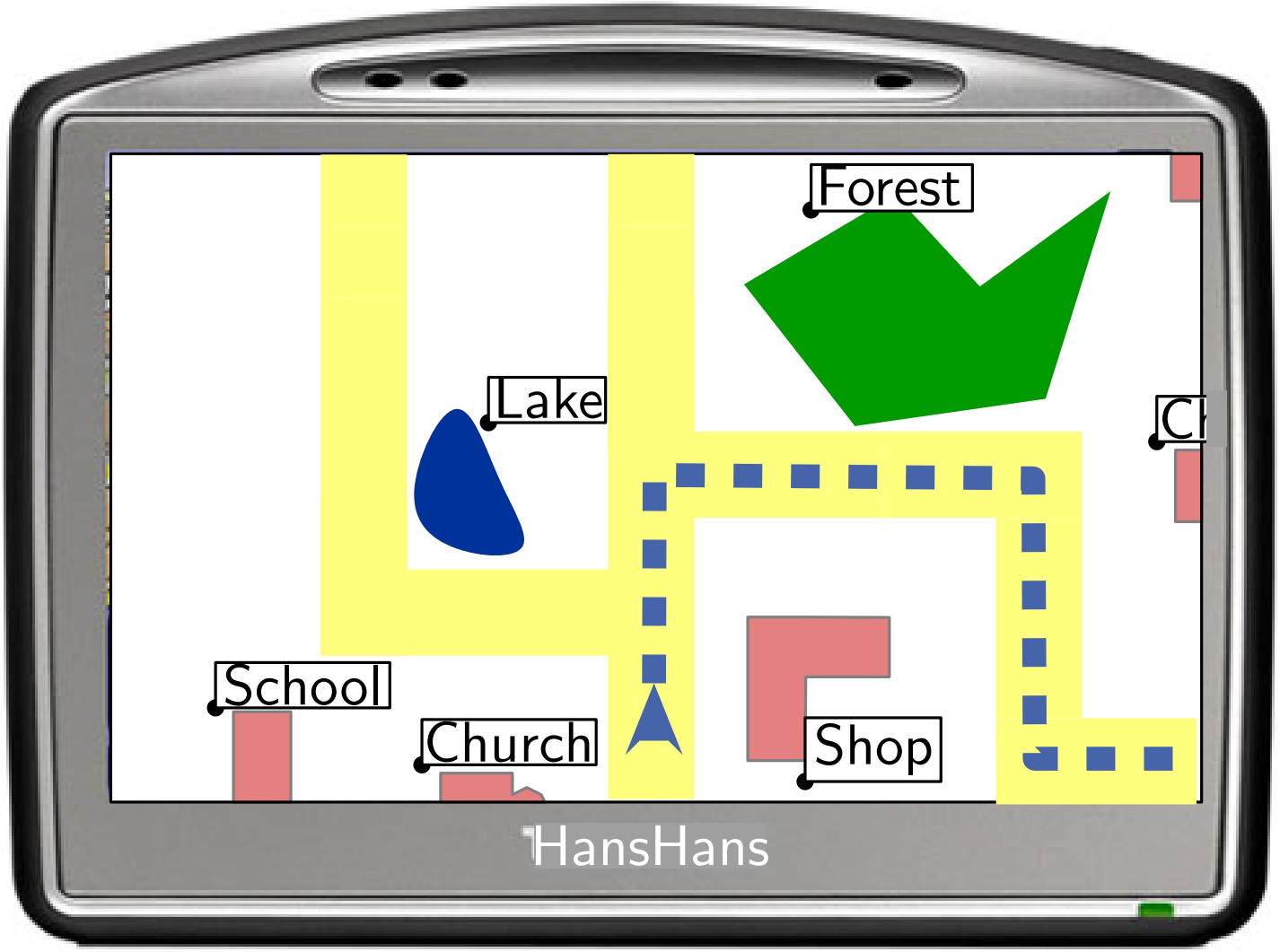
Motivation



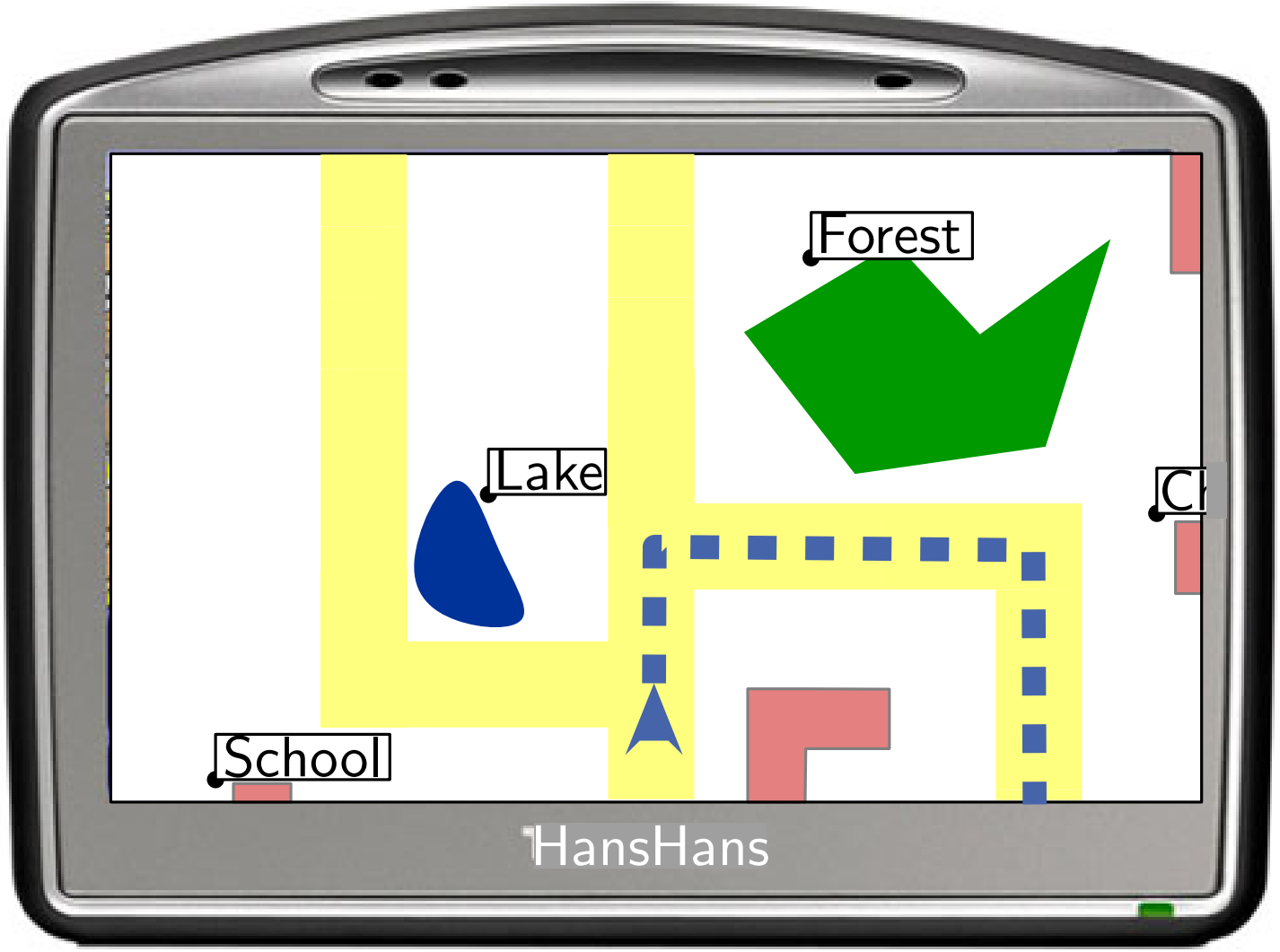
Motivation



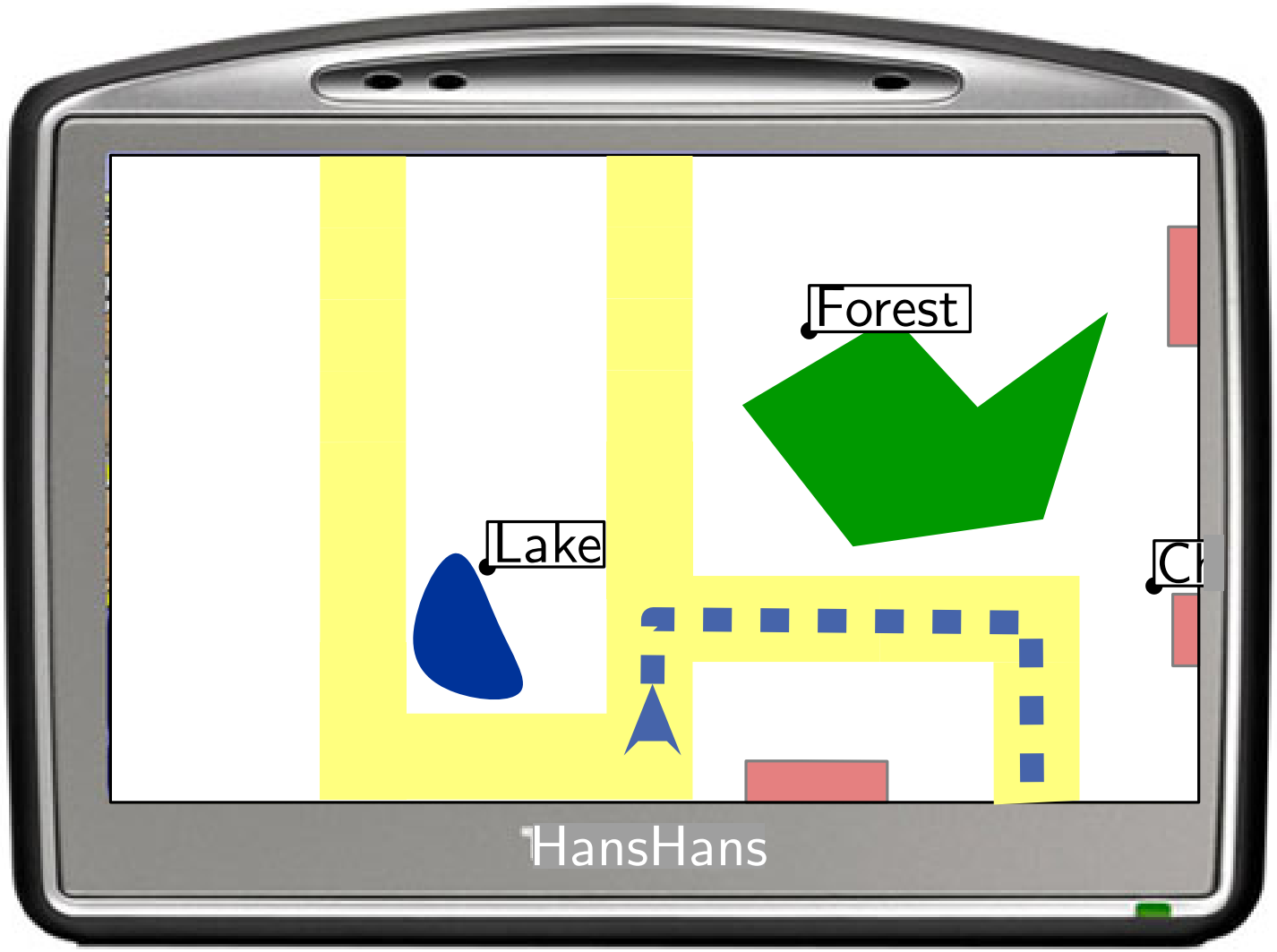
Motivation



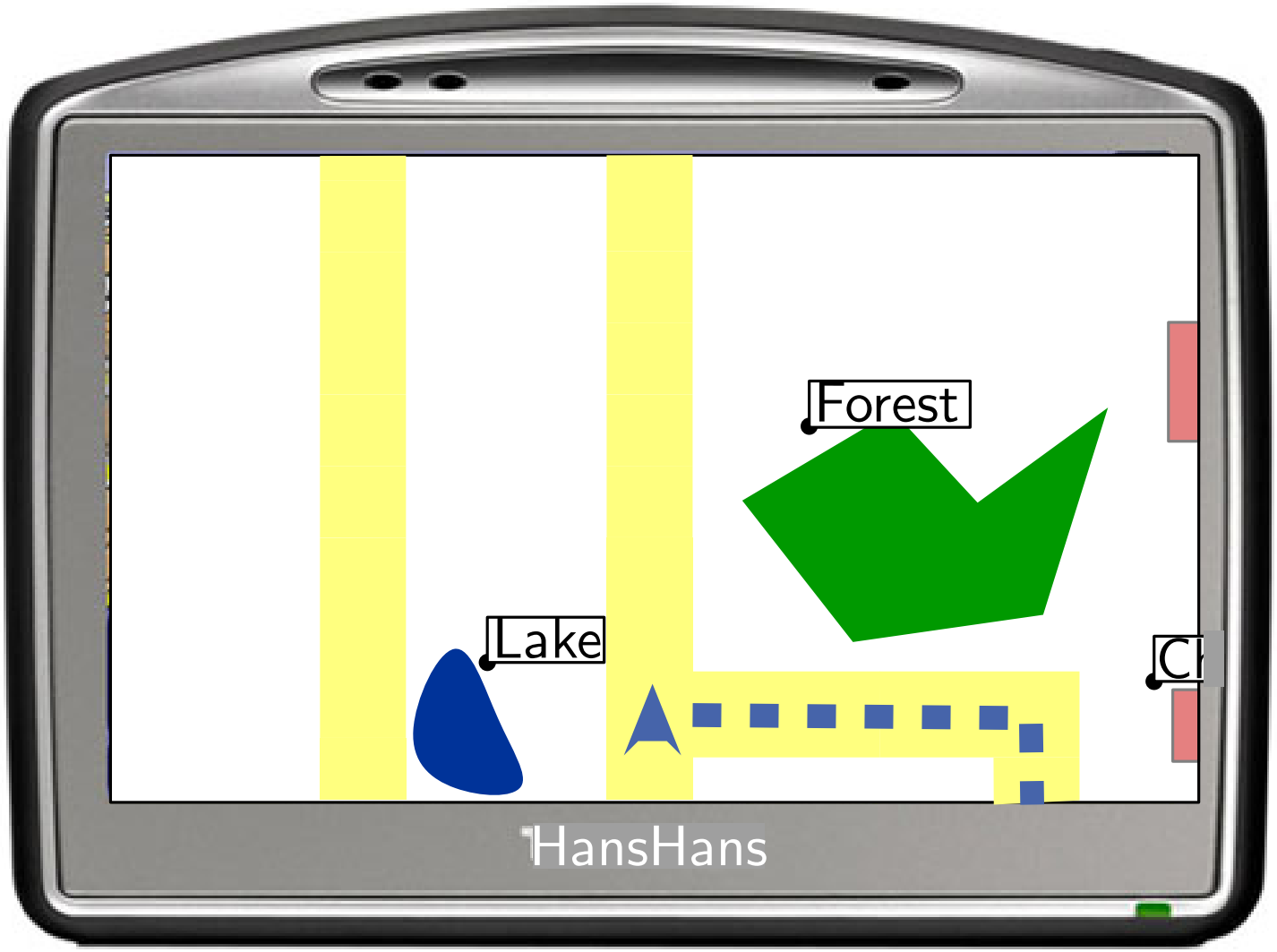
Motivation



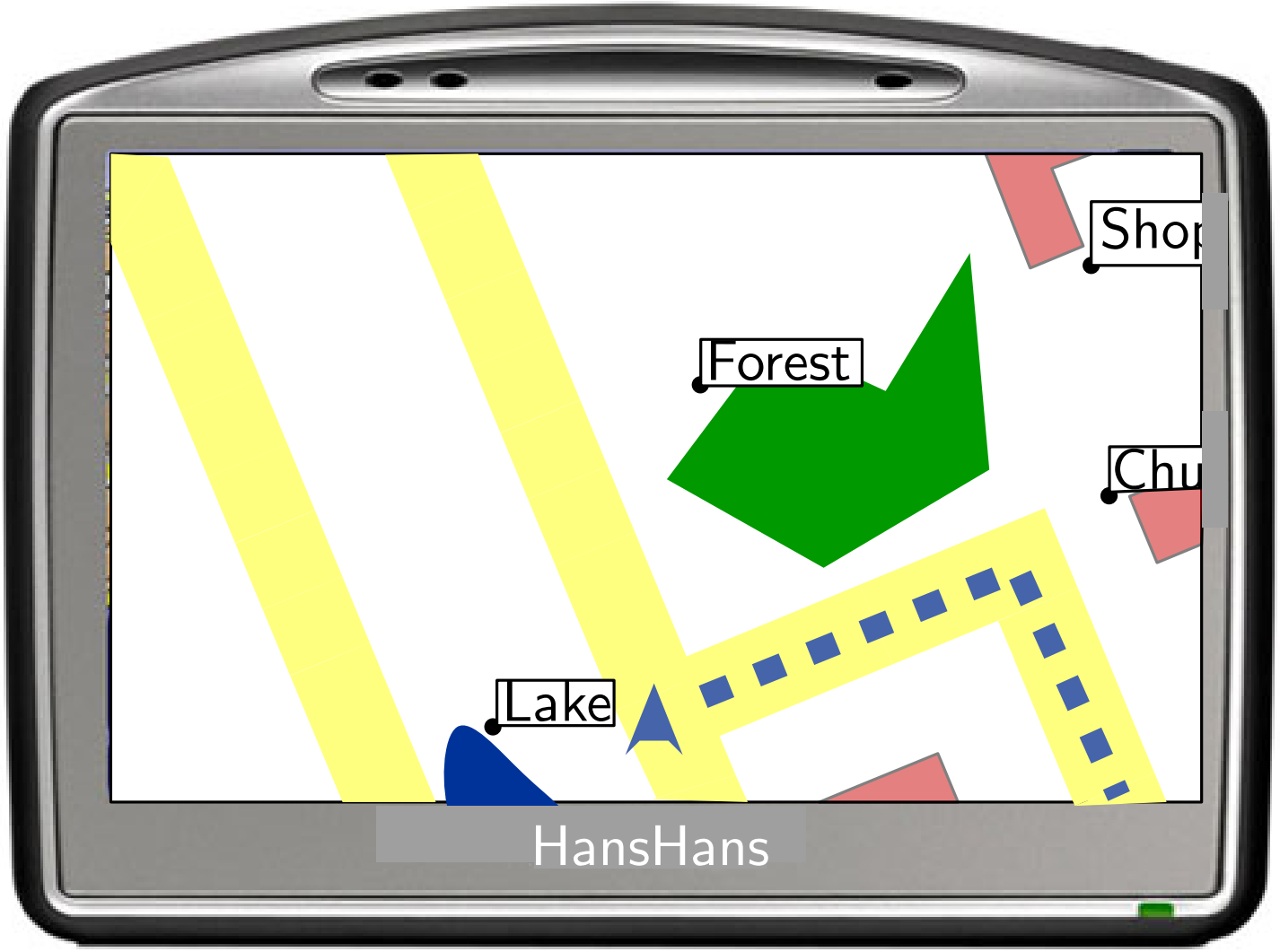
Motivation



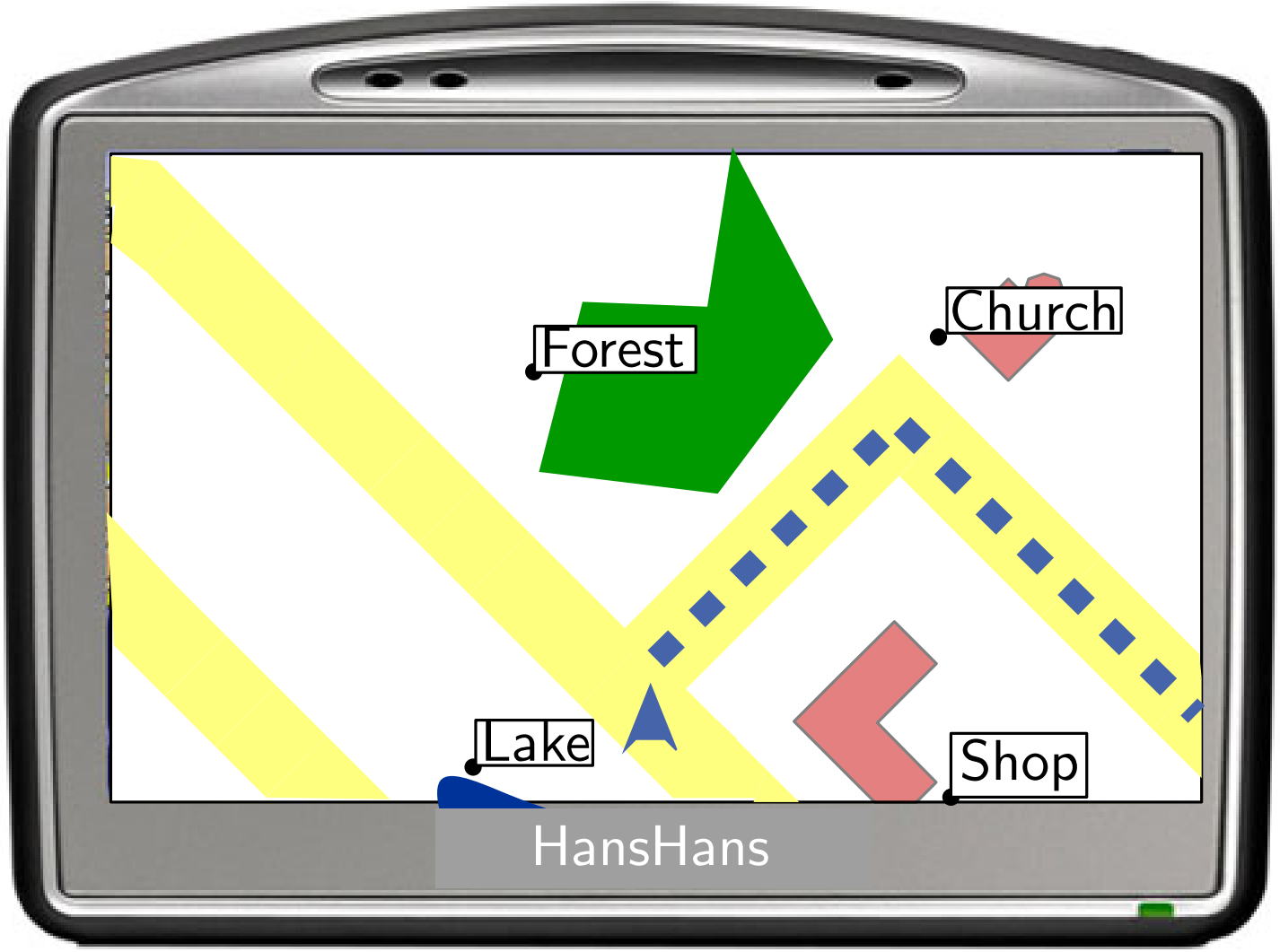
Motivation



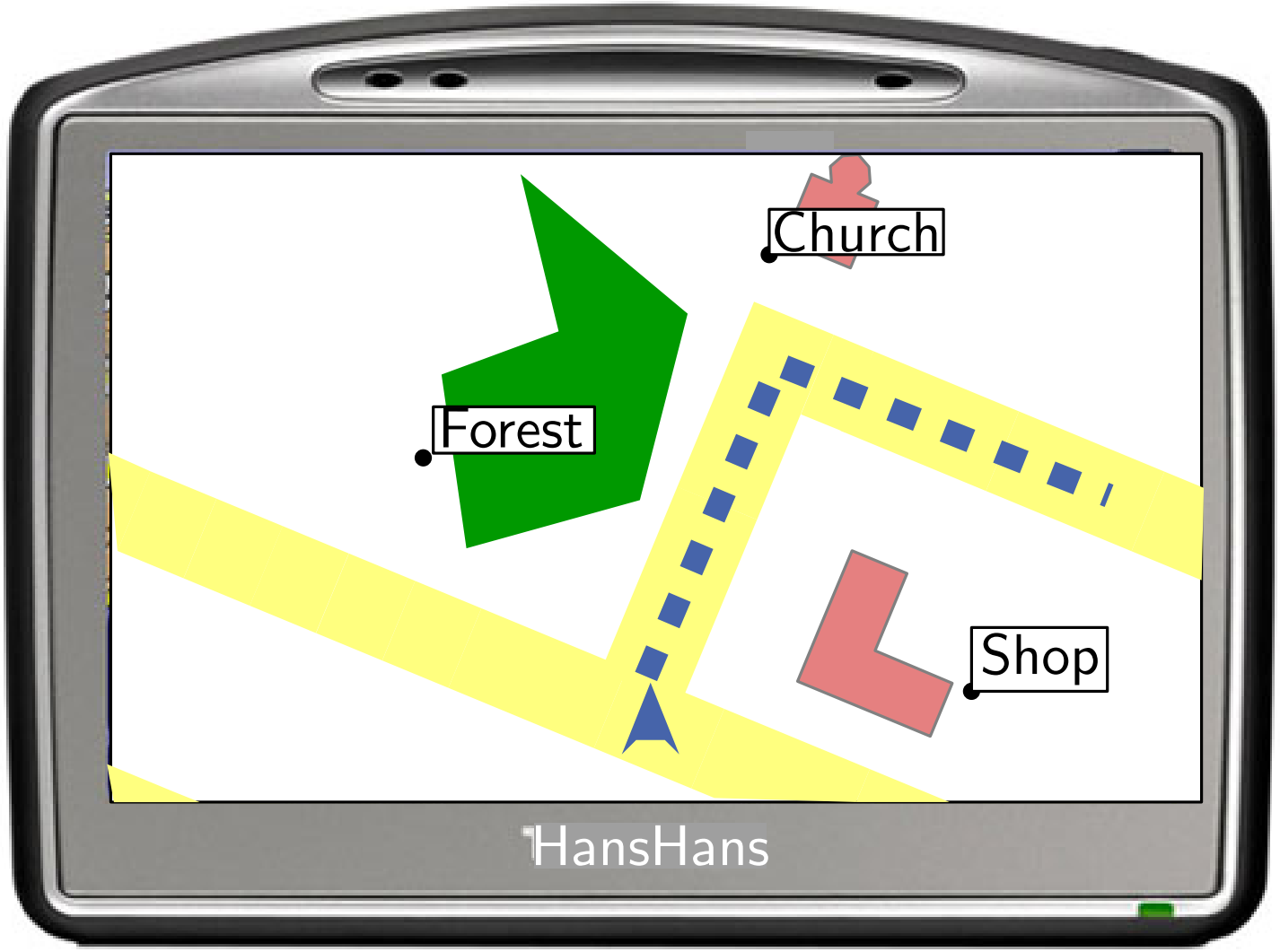
Motivation



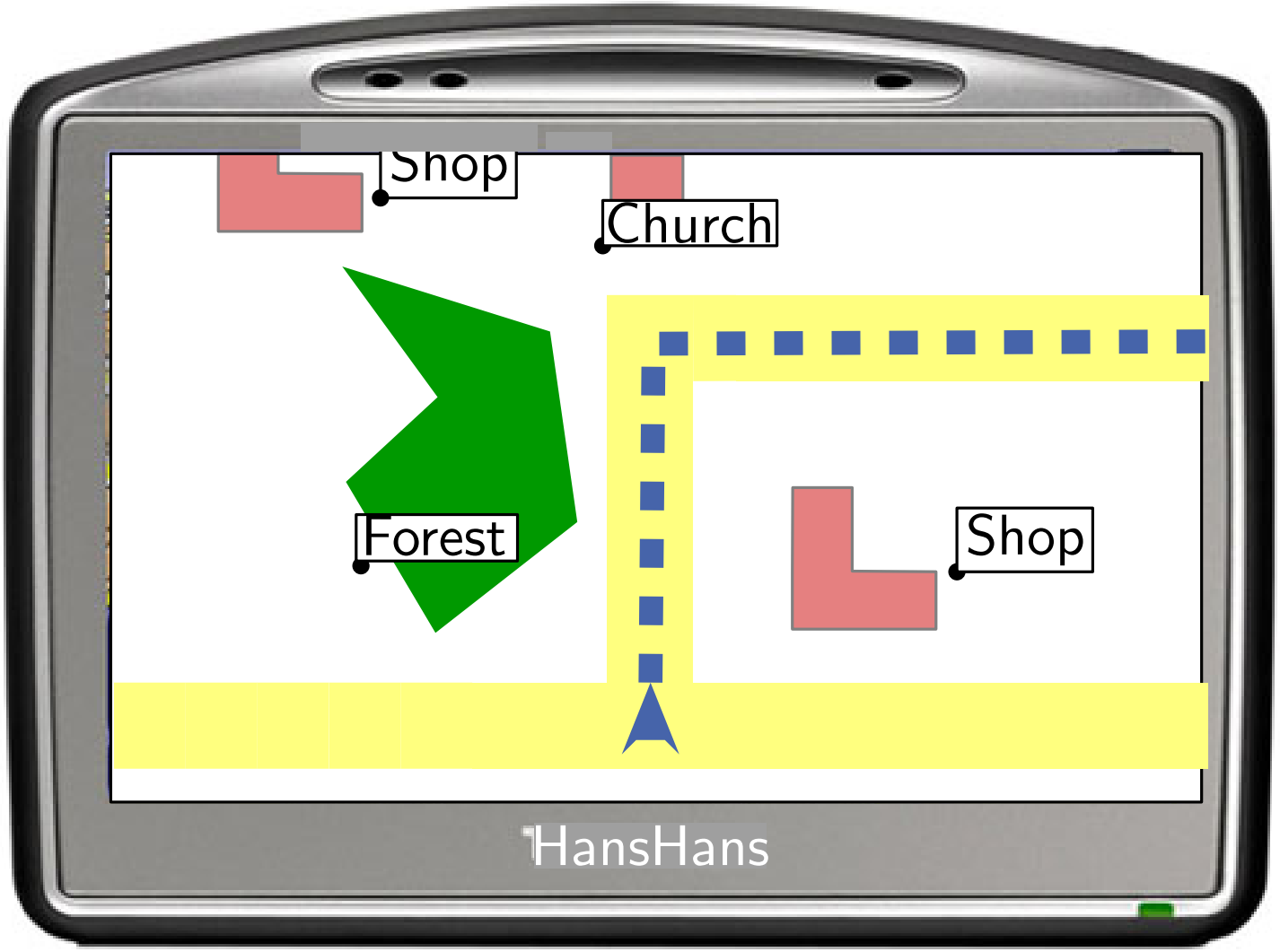
Motivation



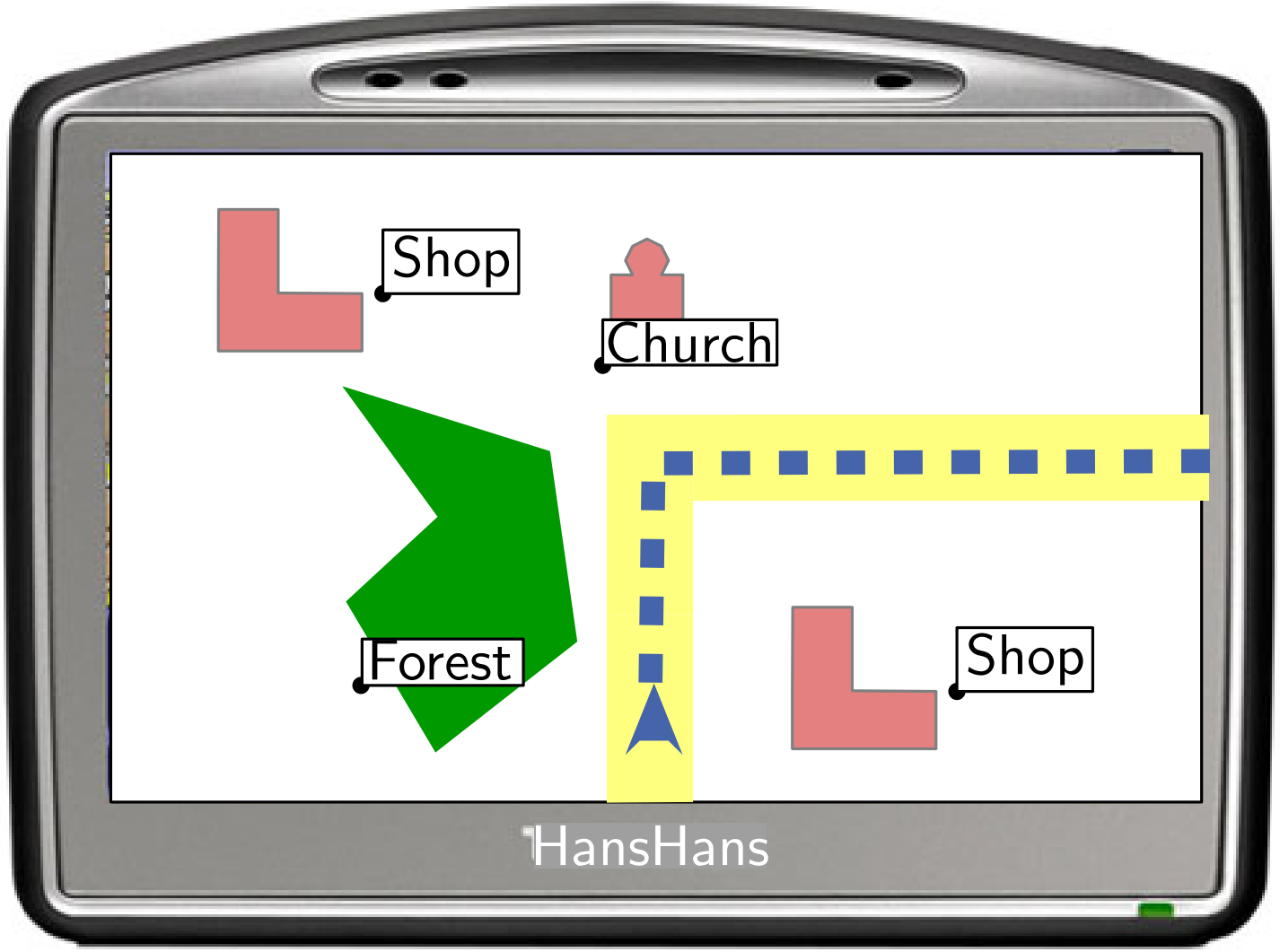
Motivation



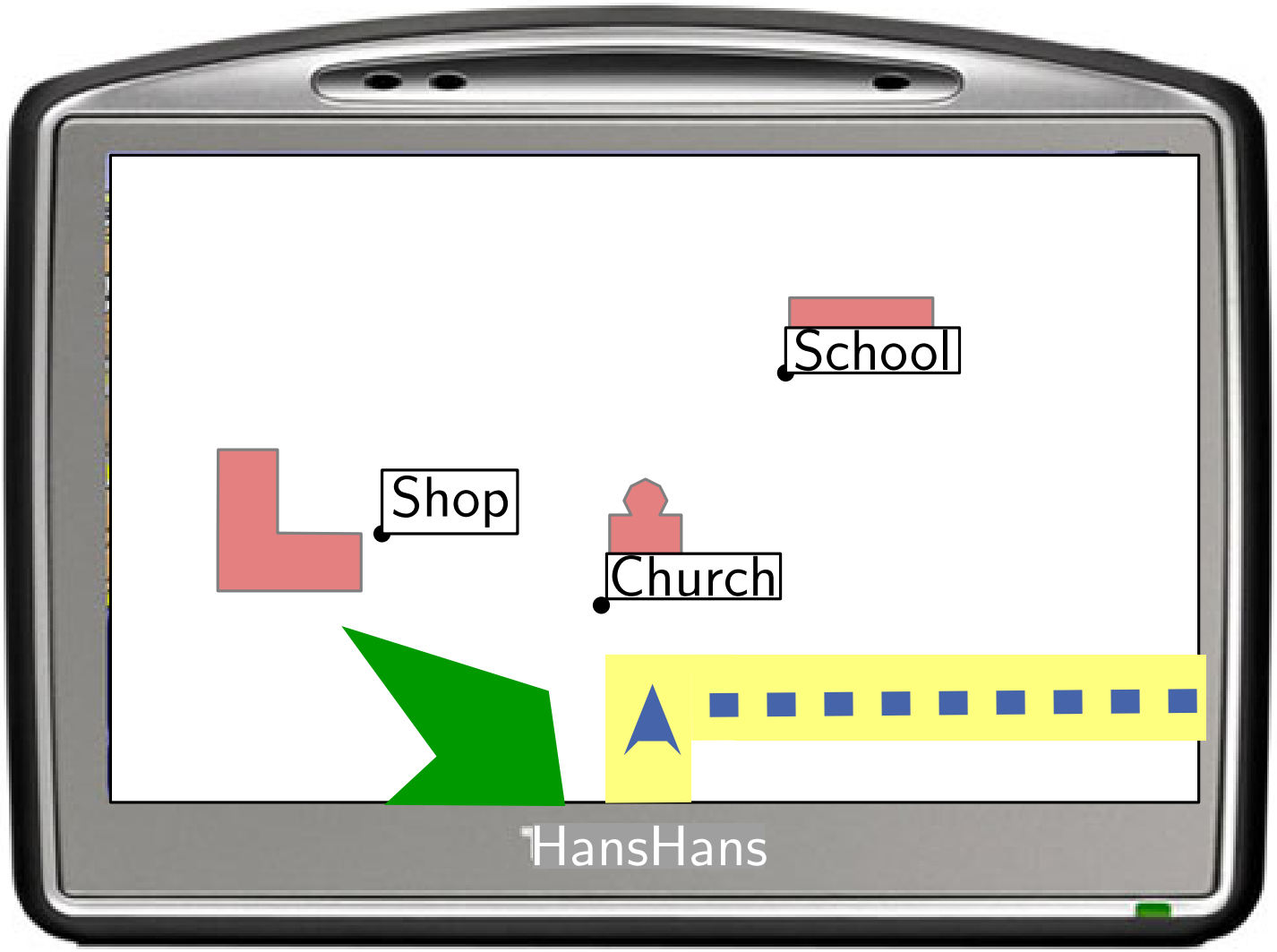
Motivation



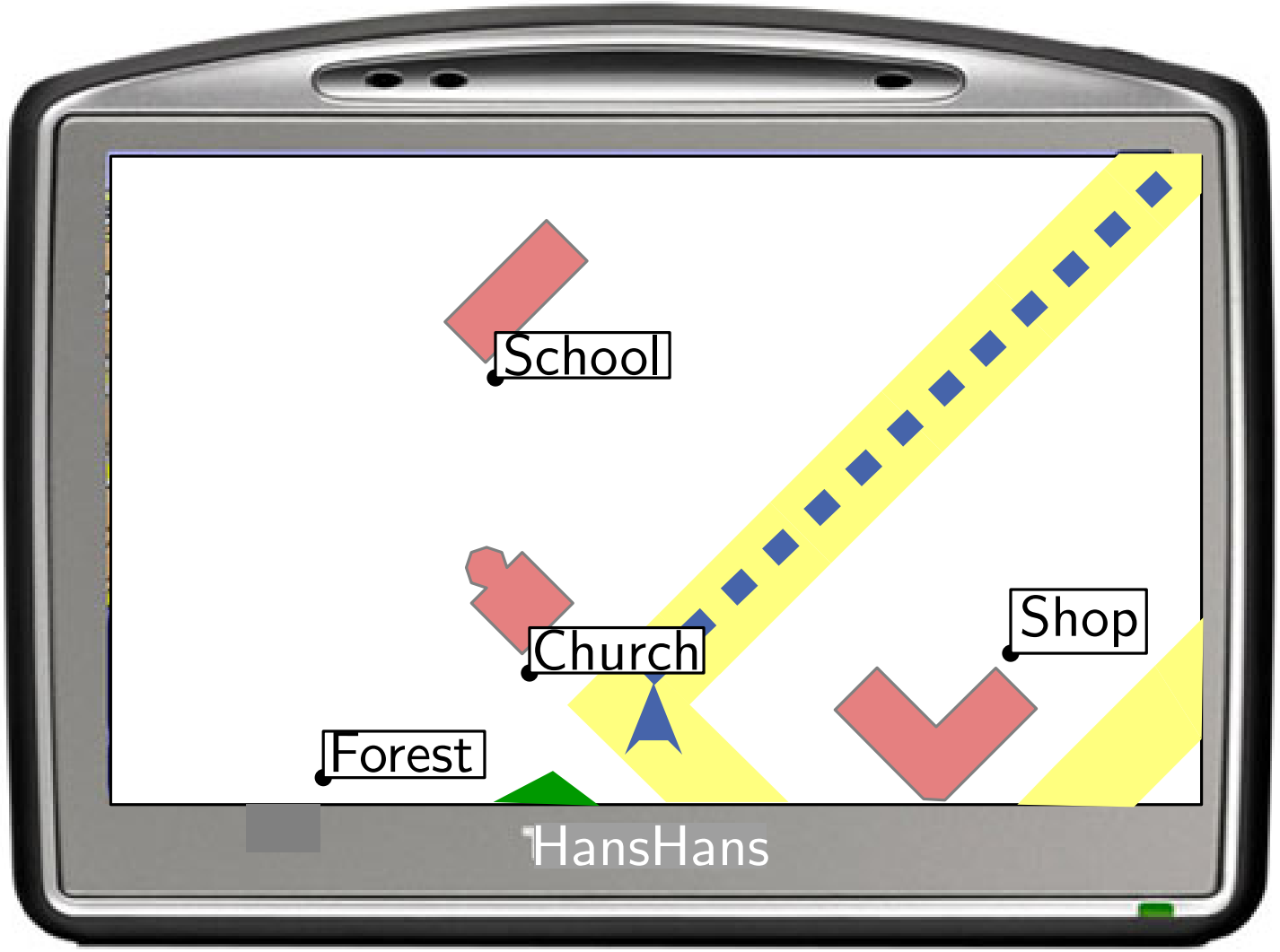
Motivation



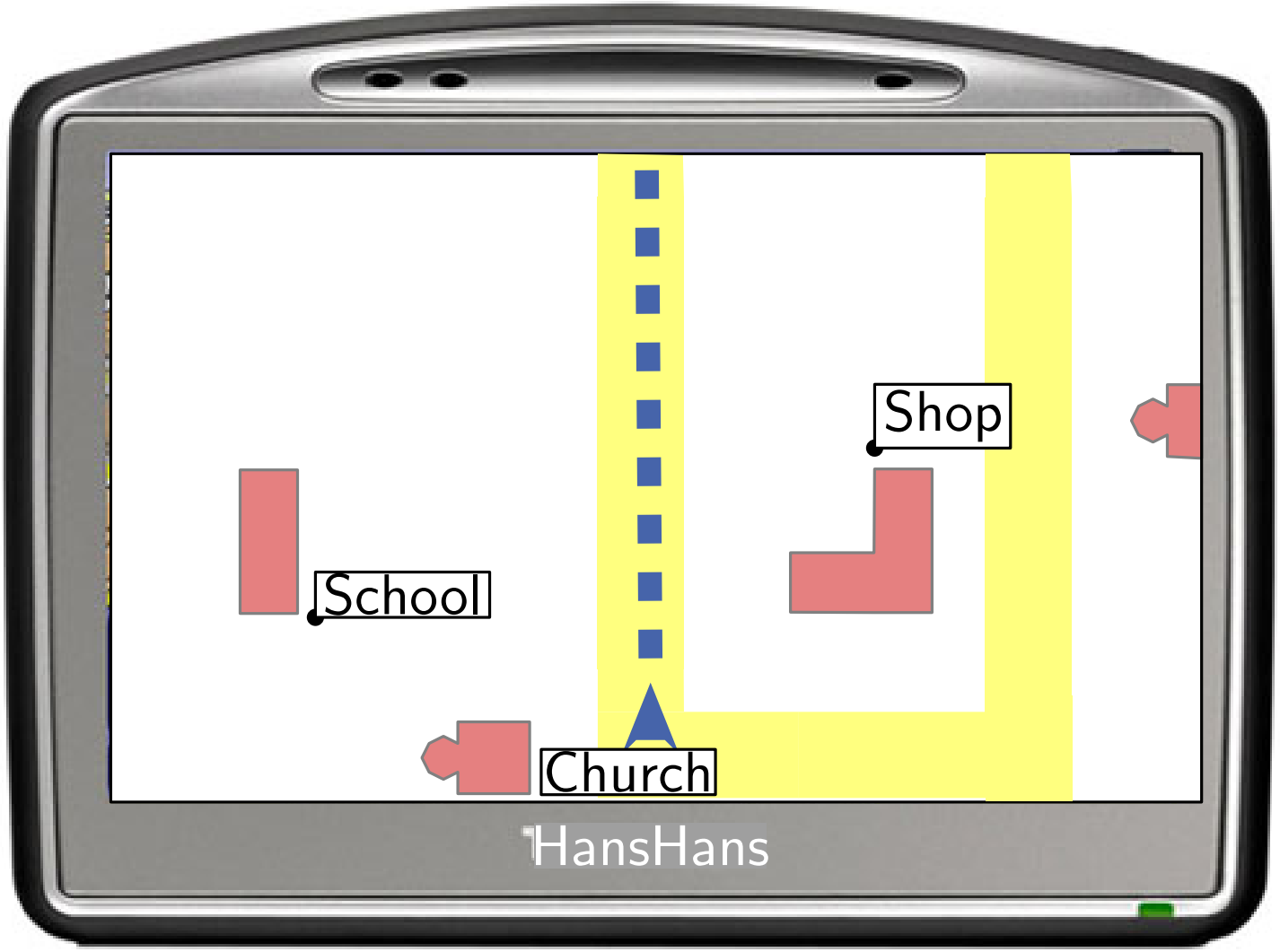
Motivation



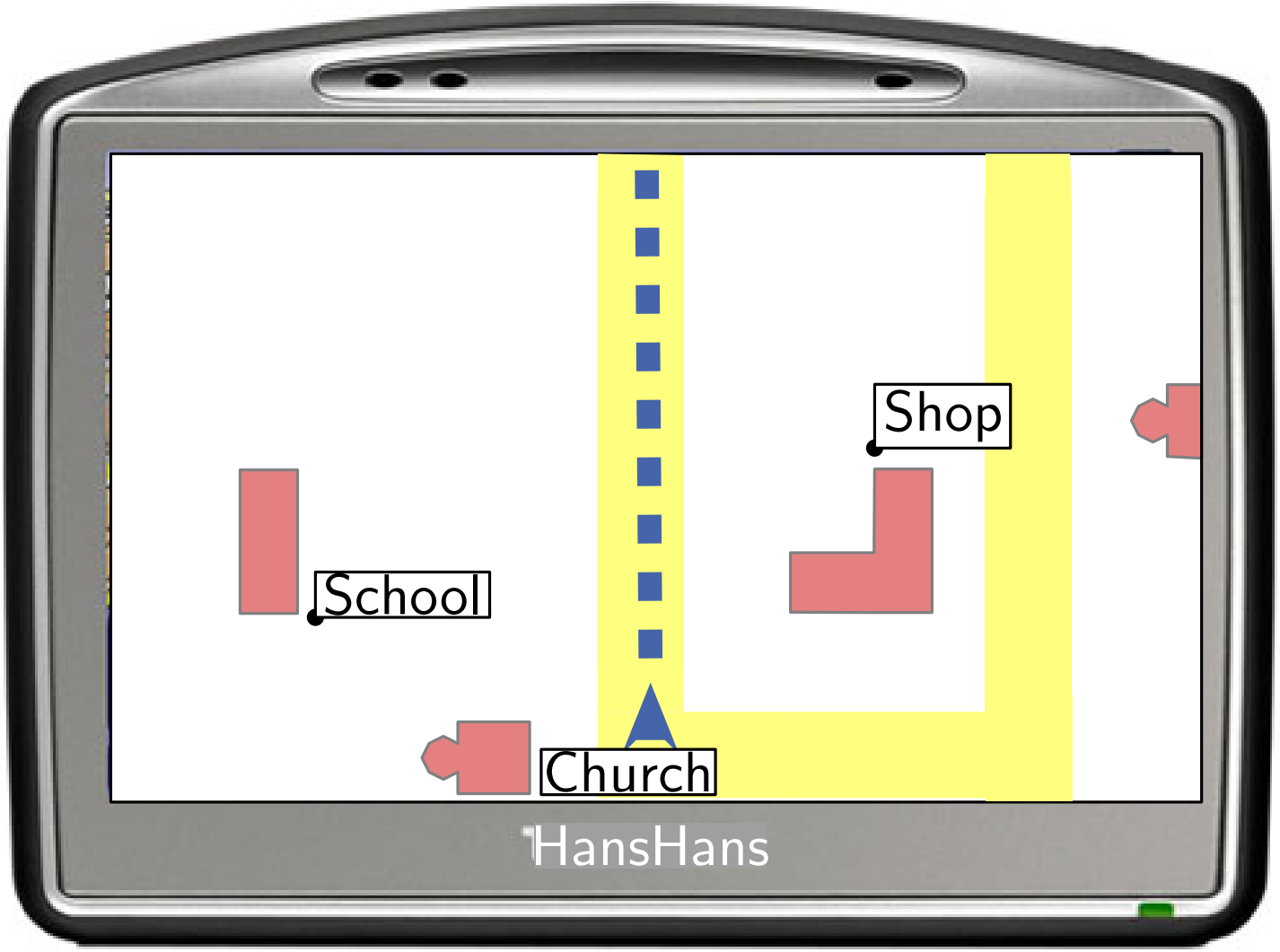
Motivation



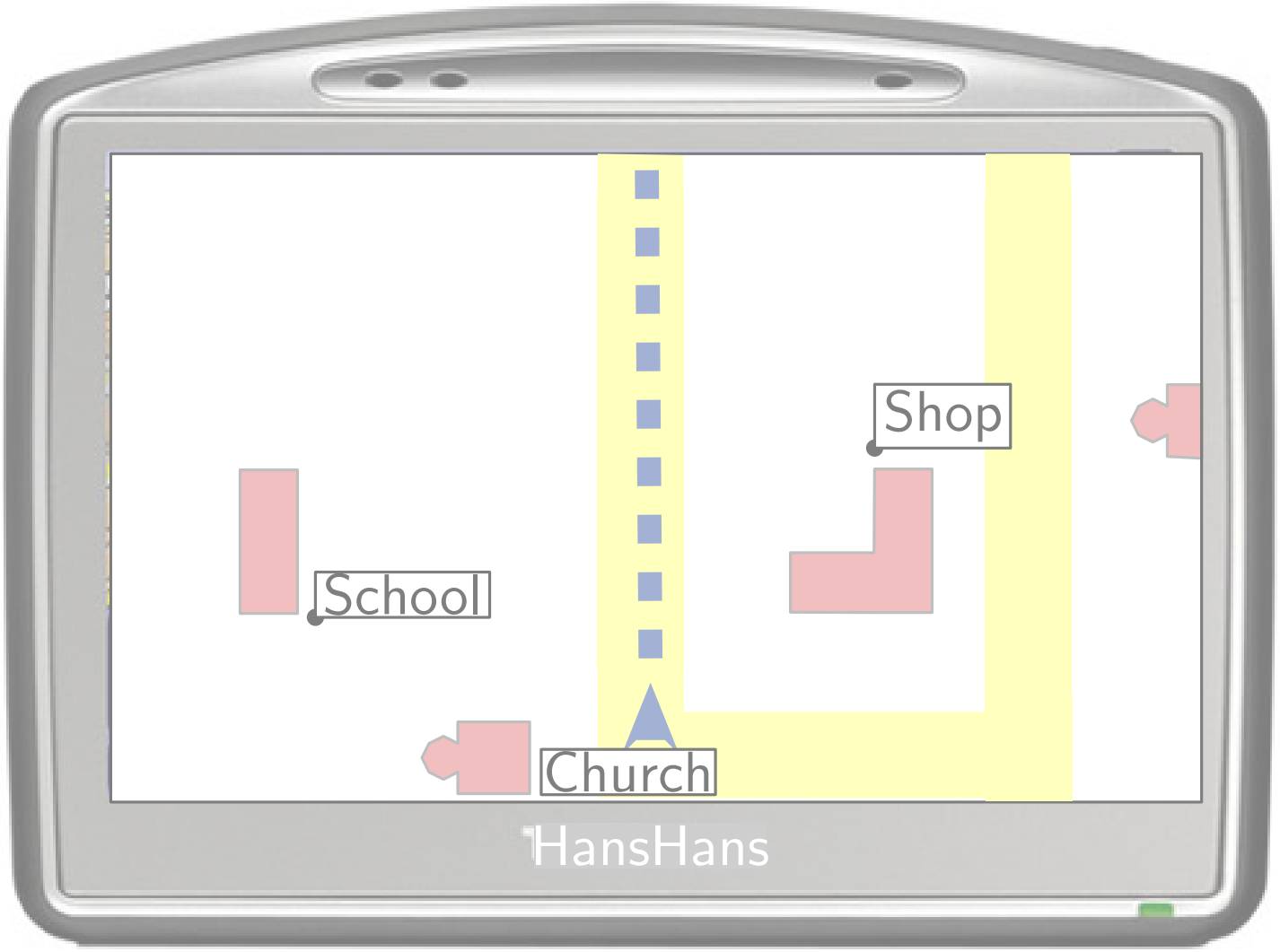
Motivation



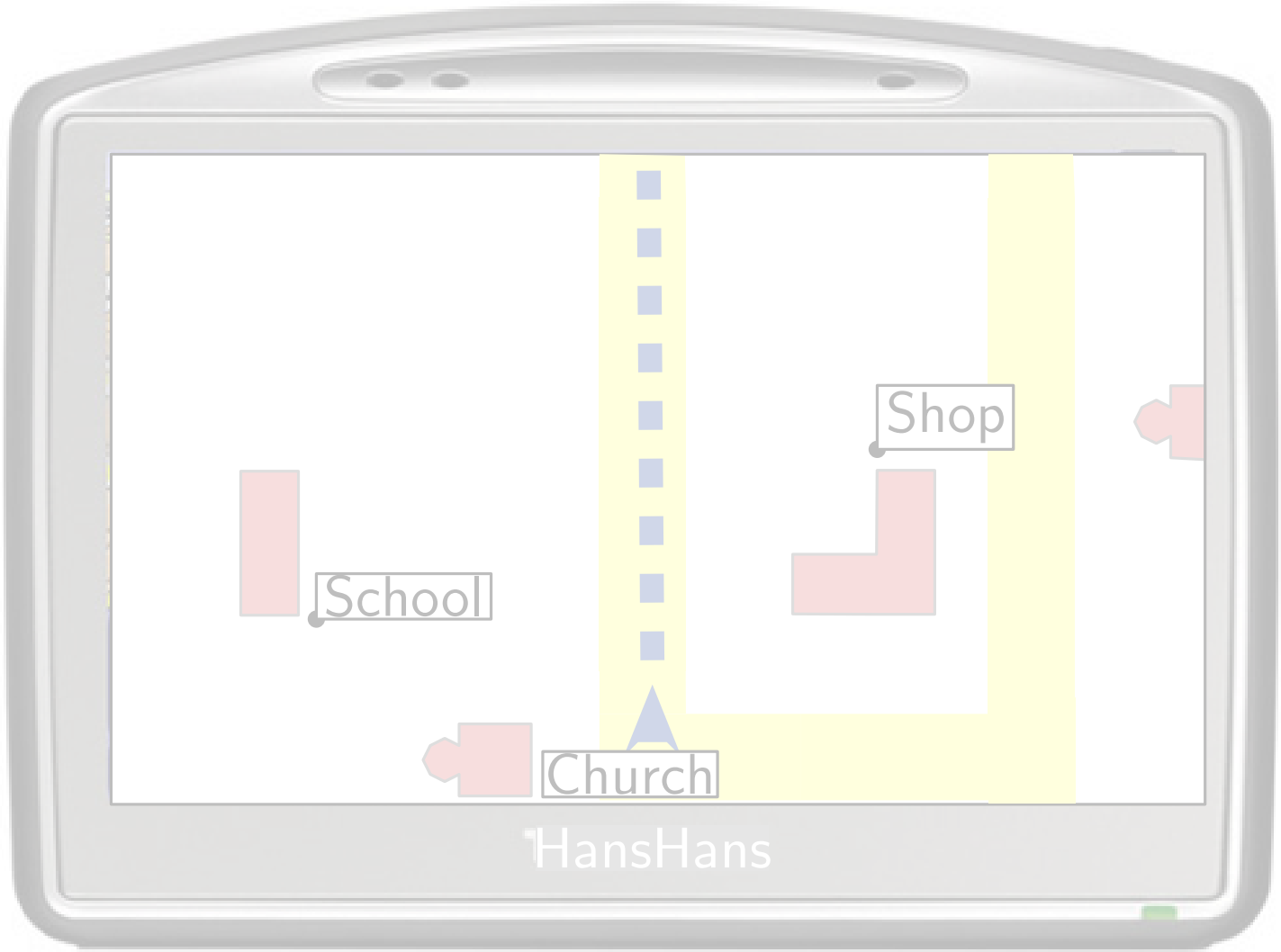
Motivation



Motivation



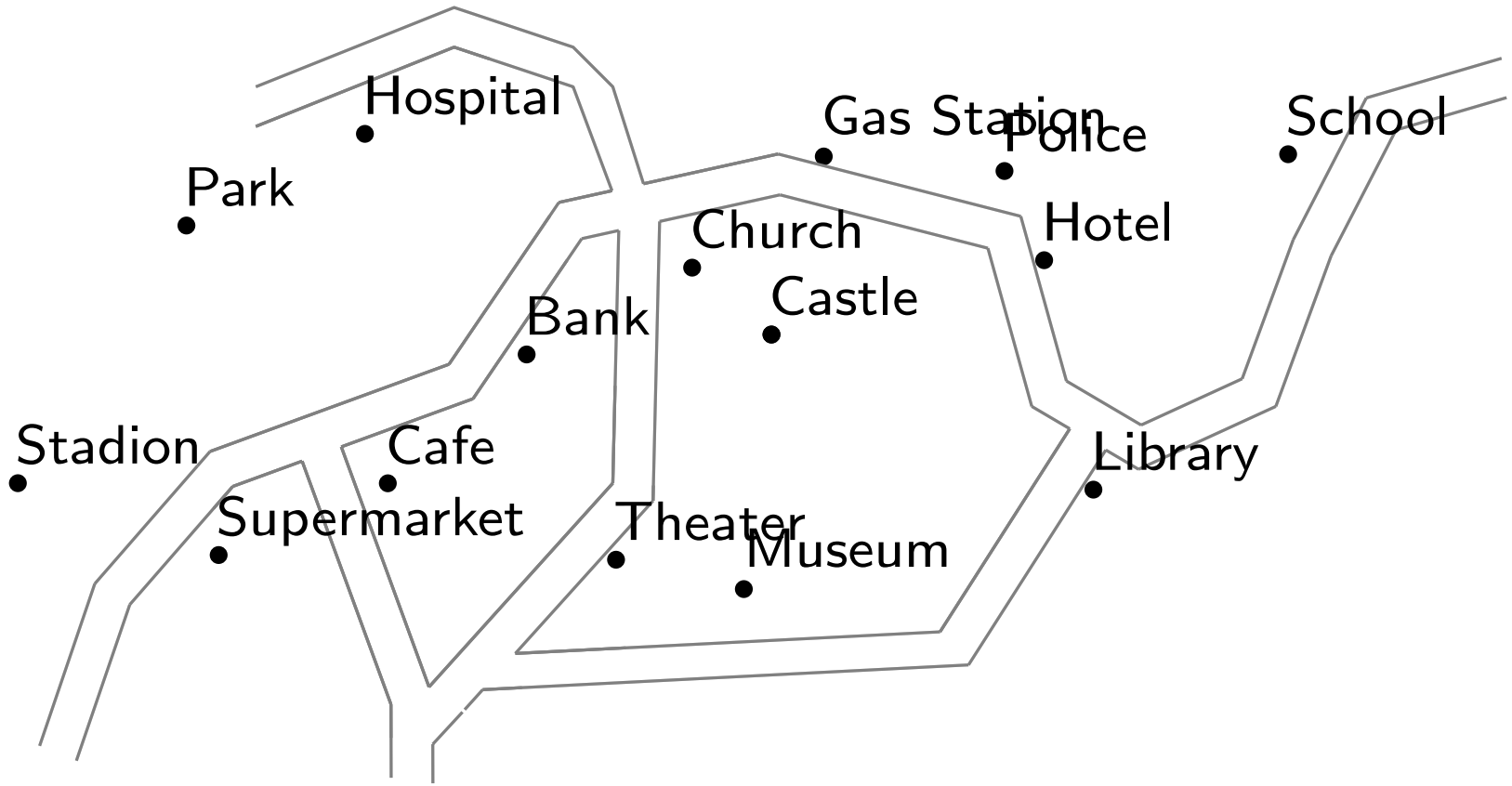
Motivation



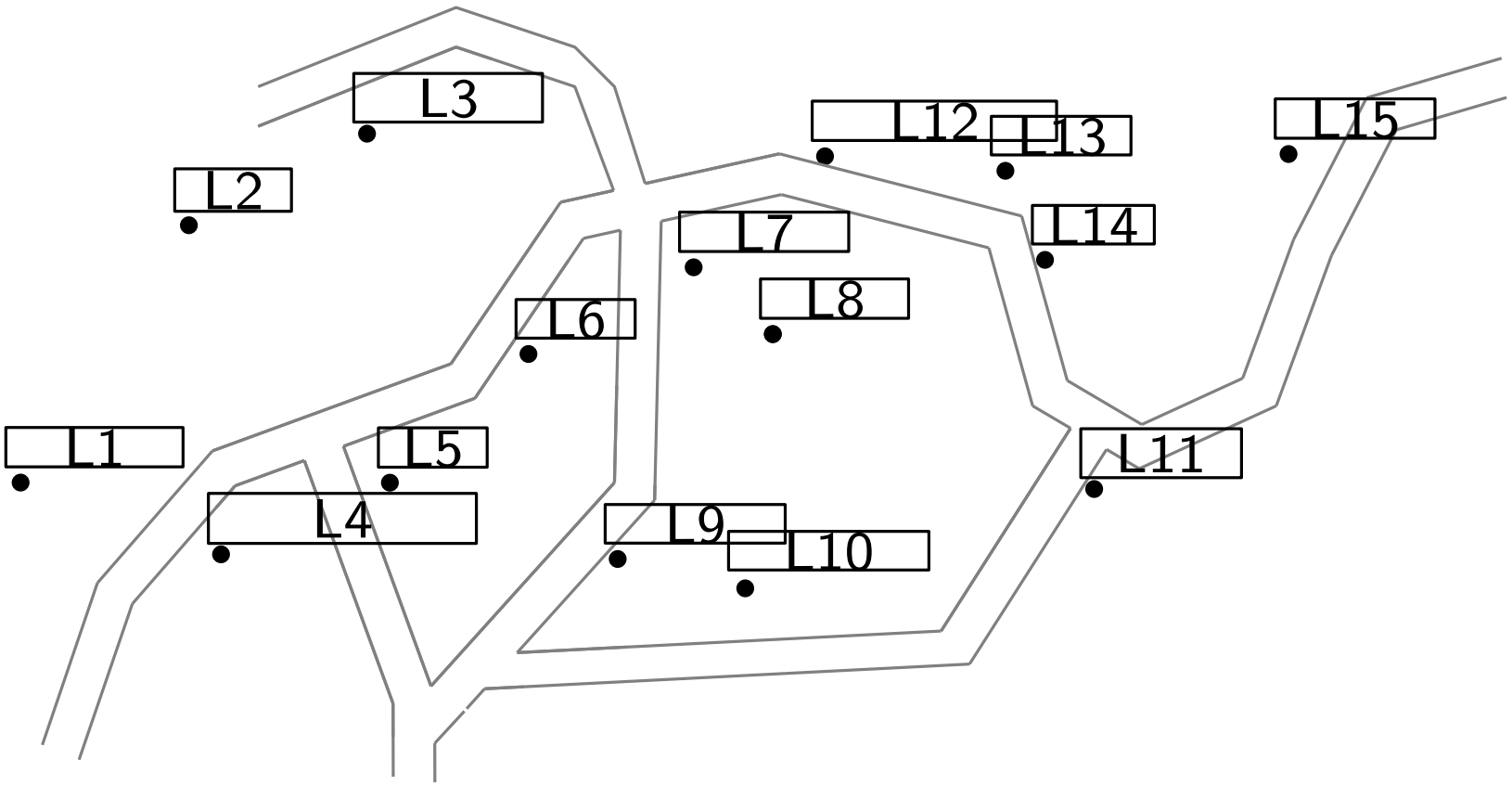
Motivation

Trajectory-Based Dynamic Map Labeling

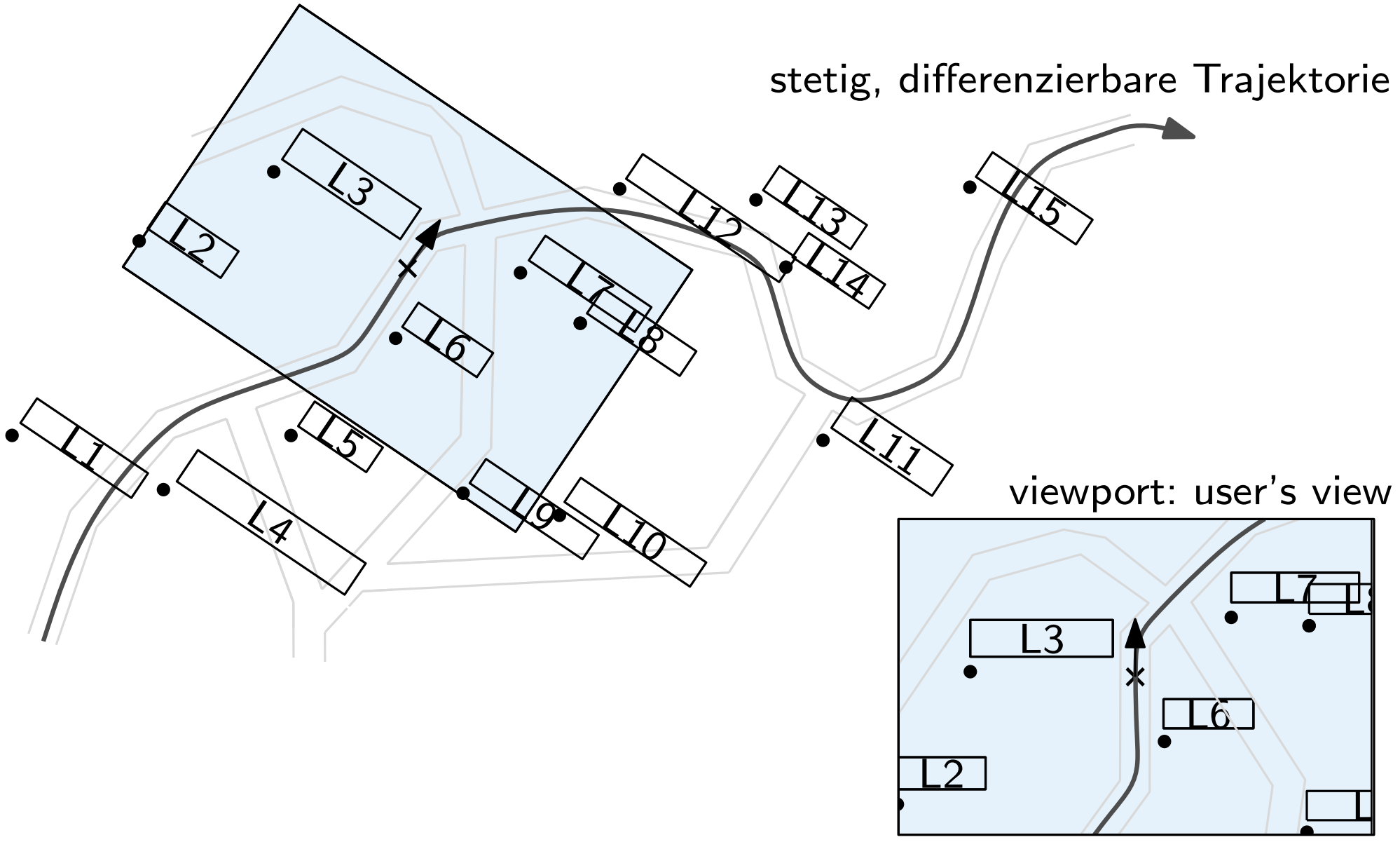
Trajectory-Based Dynamic Map Labeling



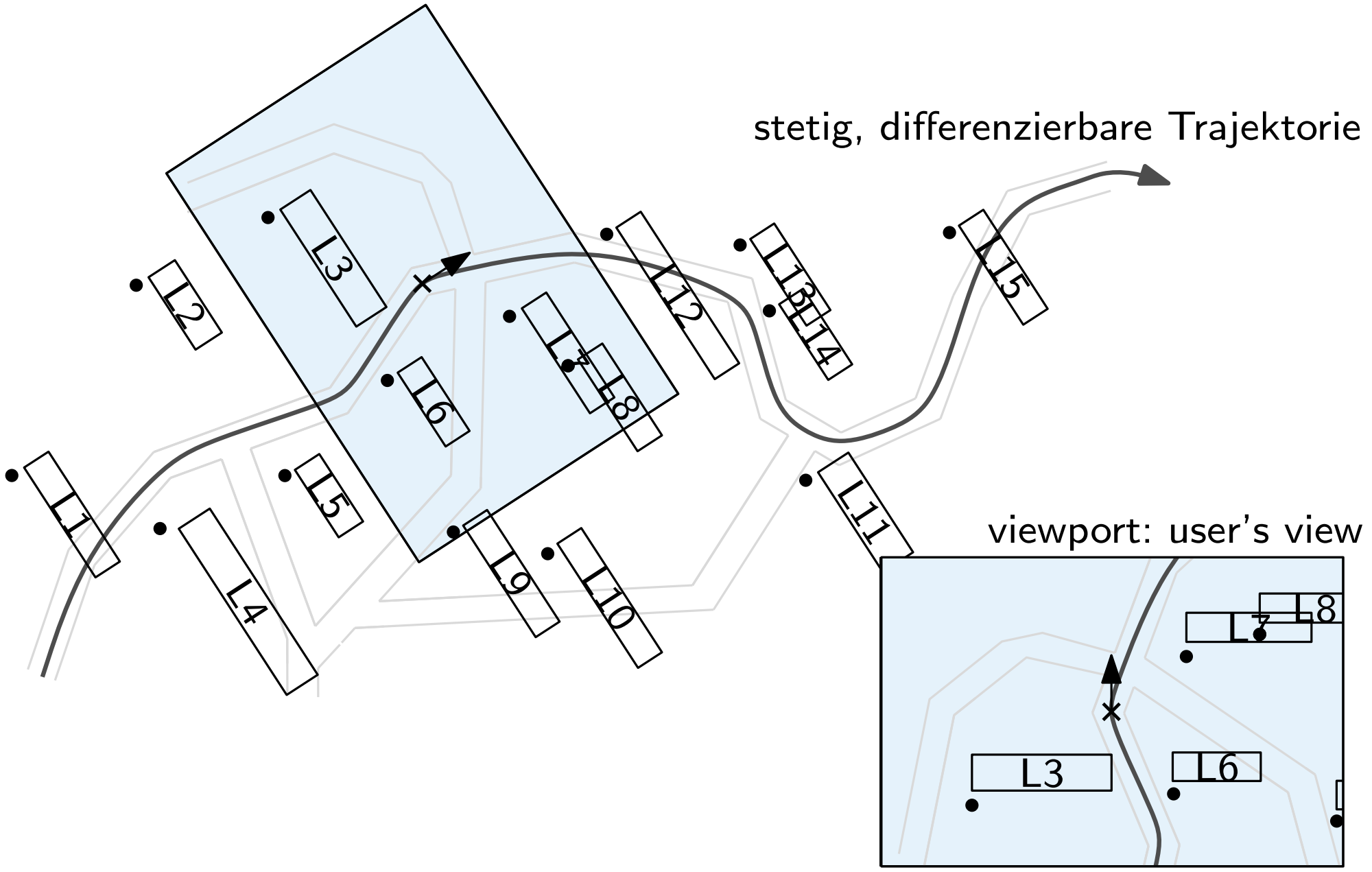
Trajectory-Based Dynamic Map Labeling



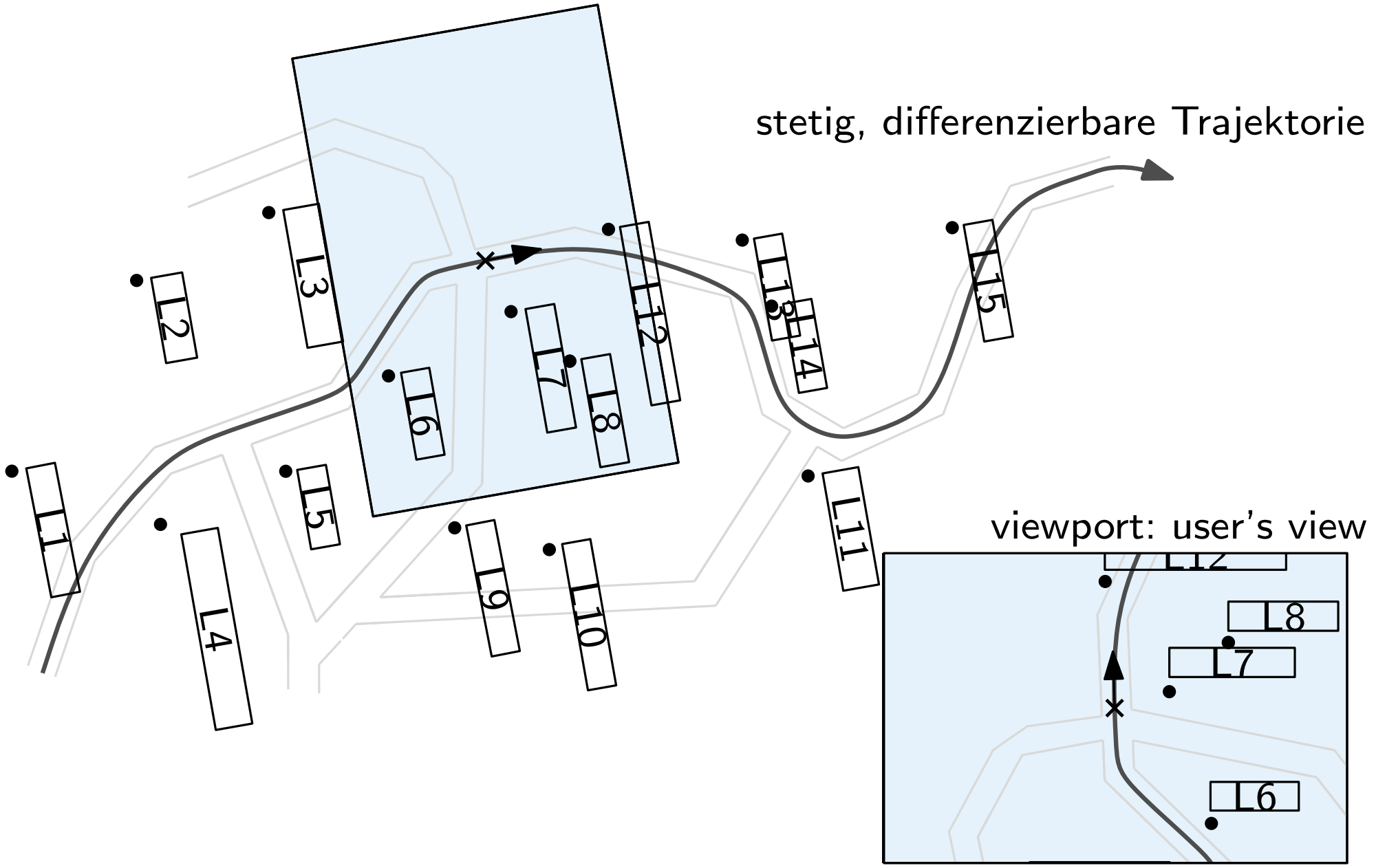
Trajectory-Based Dynamic Map Labeling



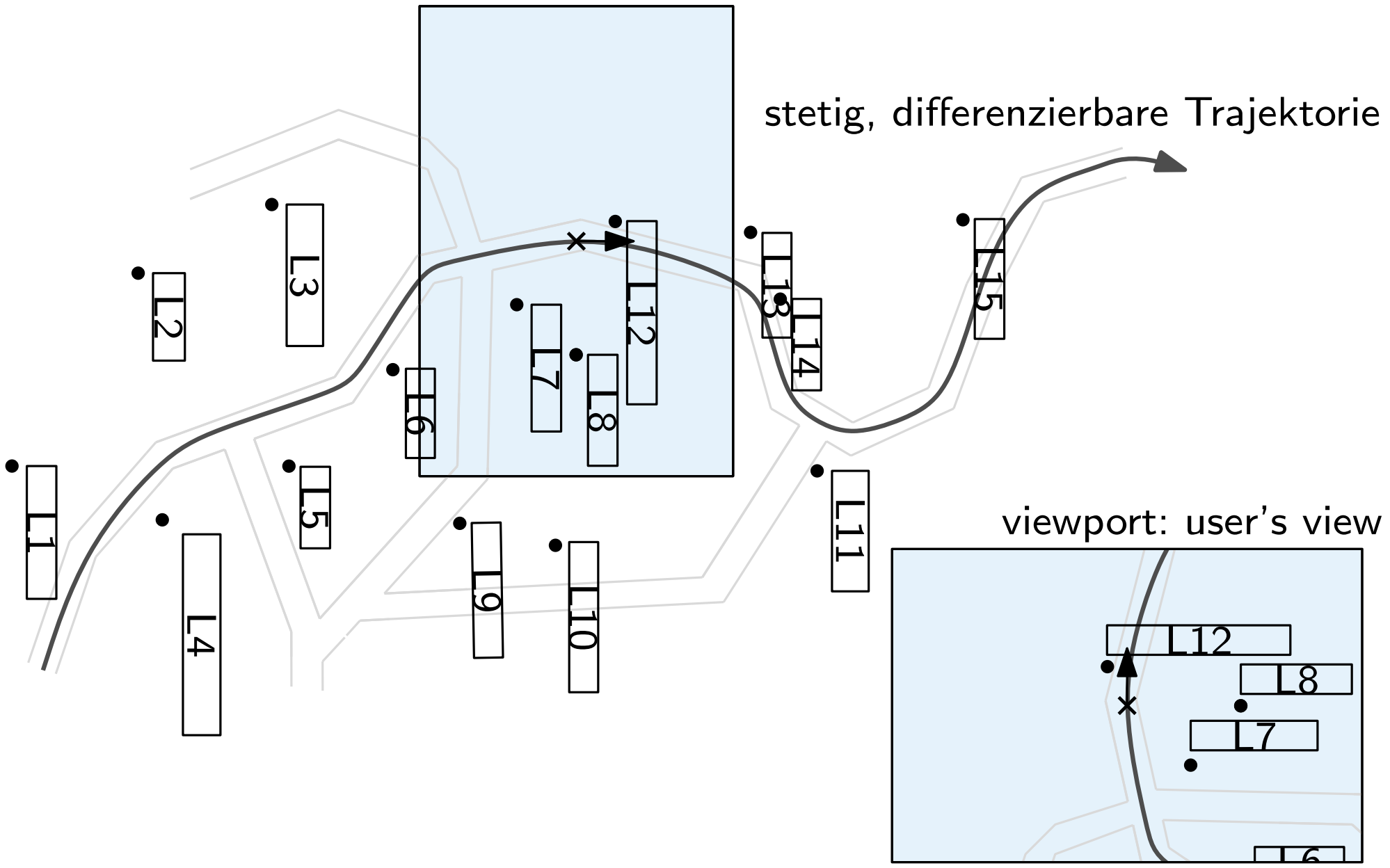
Trajectory-Based Dynamic Map Labeling



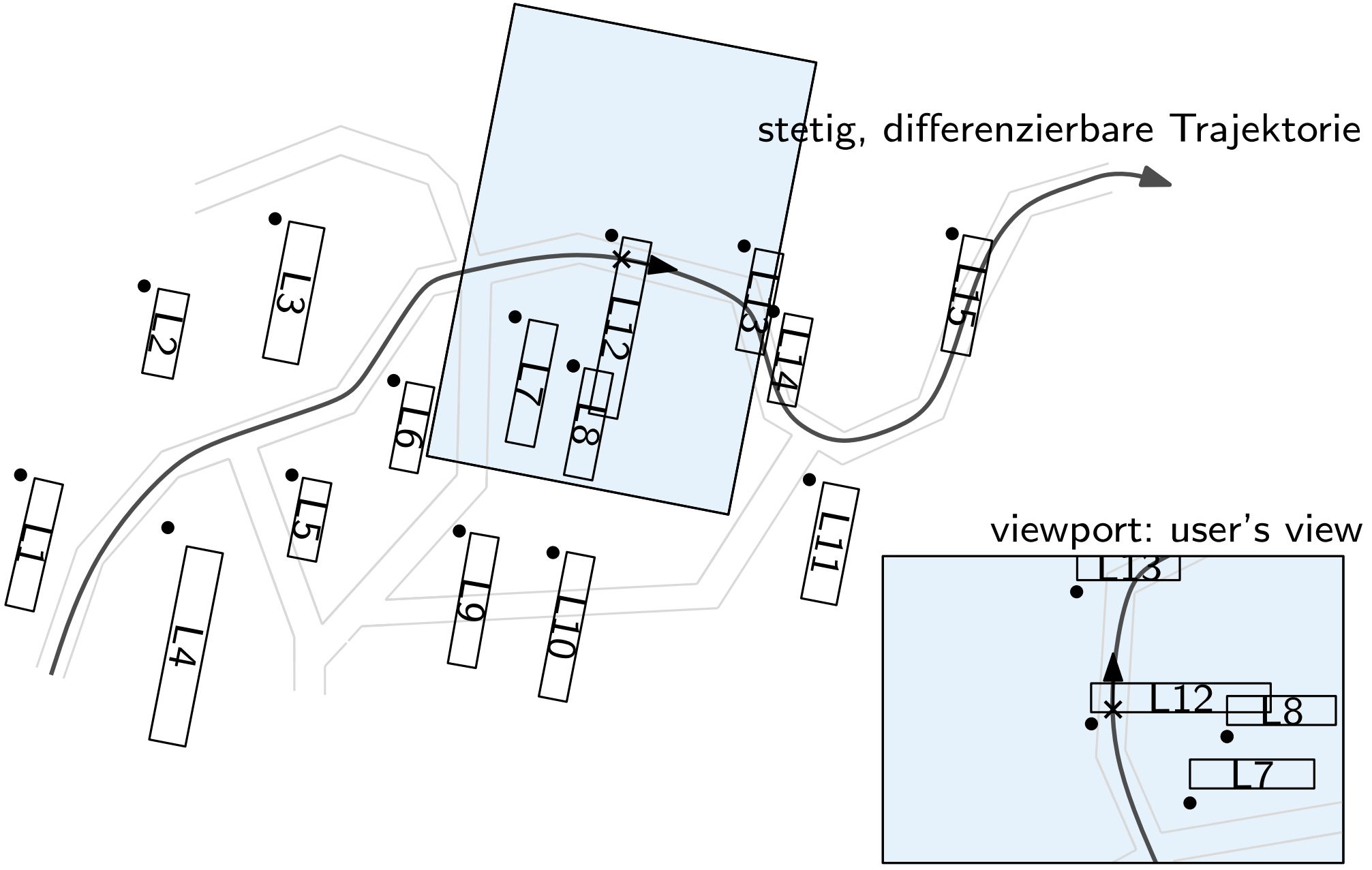
Trajectory-Based Dynamic Map Labeling



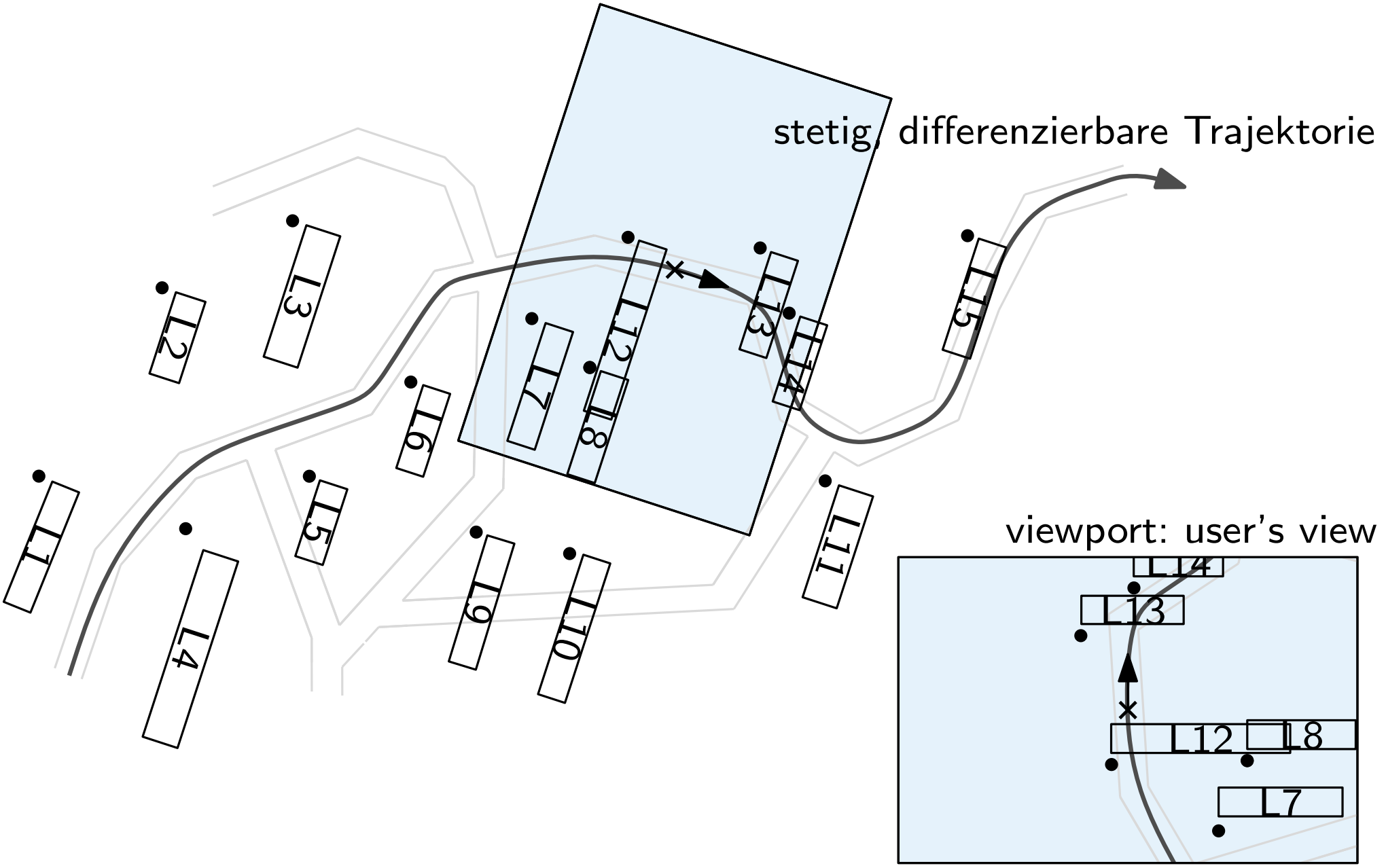
Trajectory-Based Dynamic Map Labeling



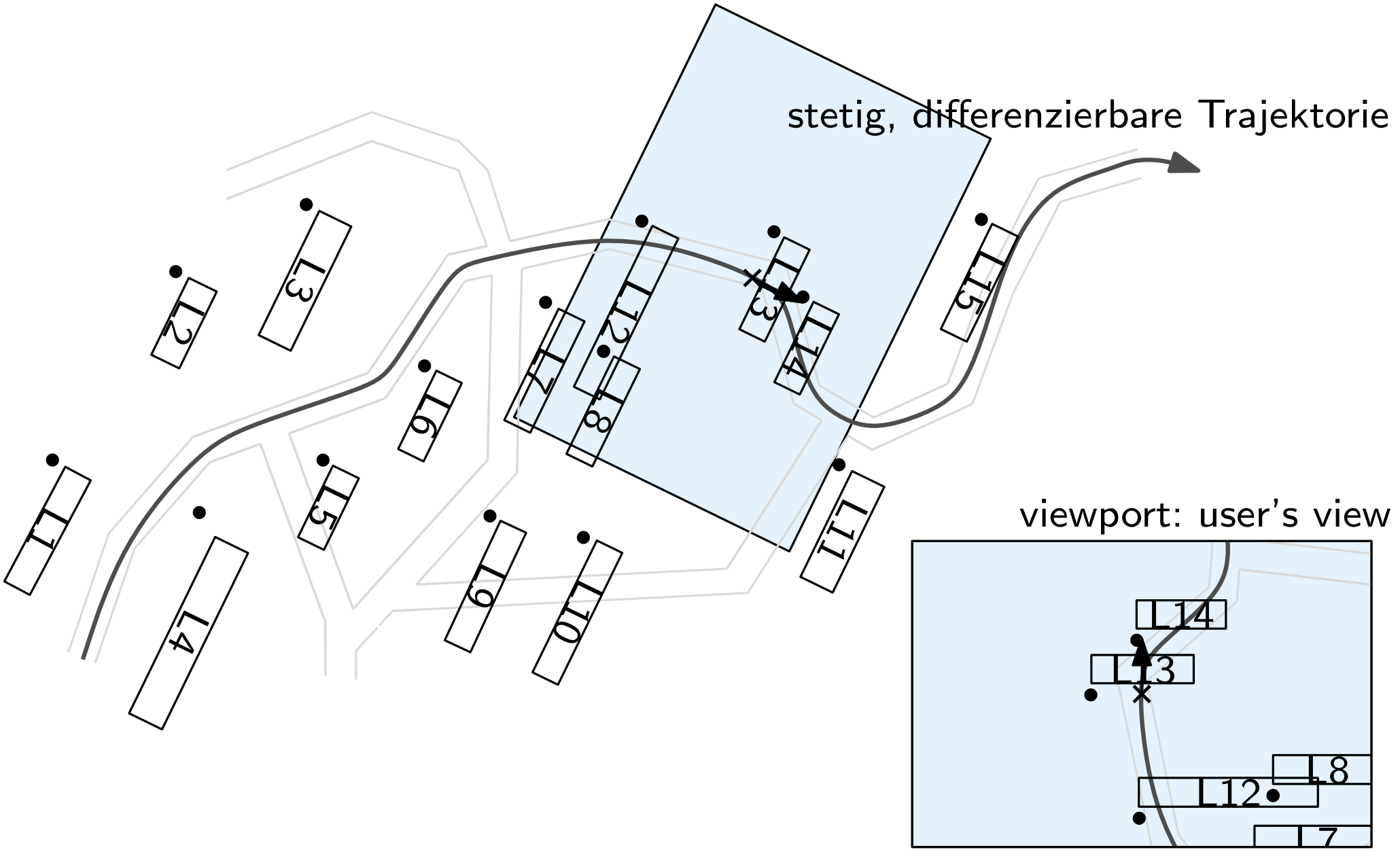
Trajectory-Based Dynamic Map Labeling



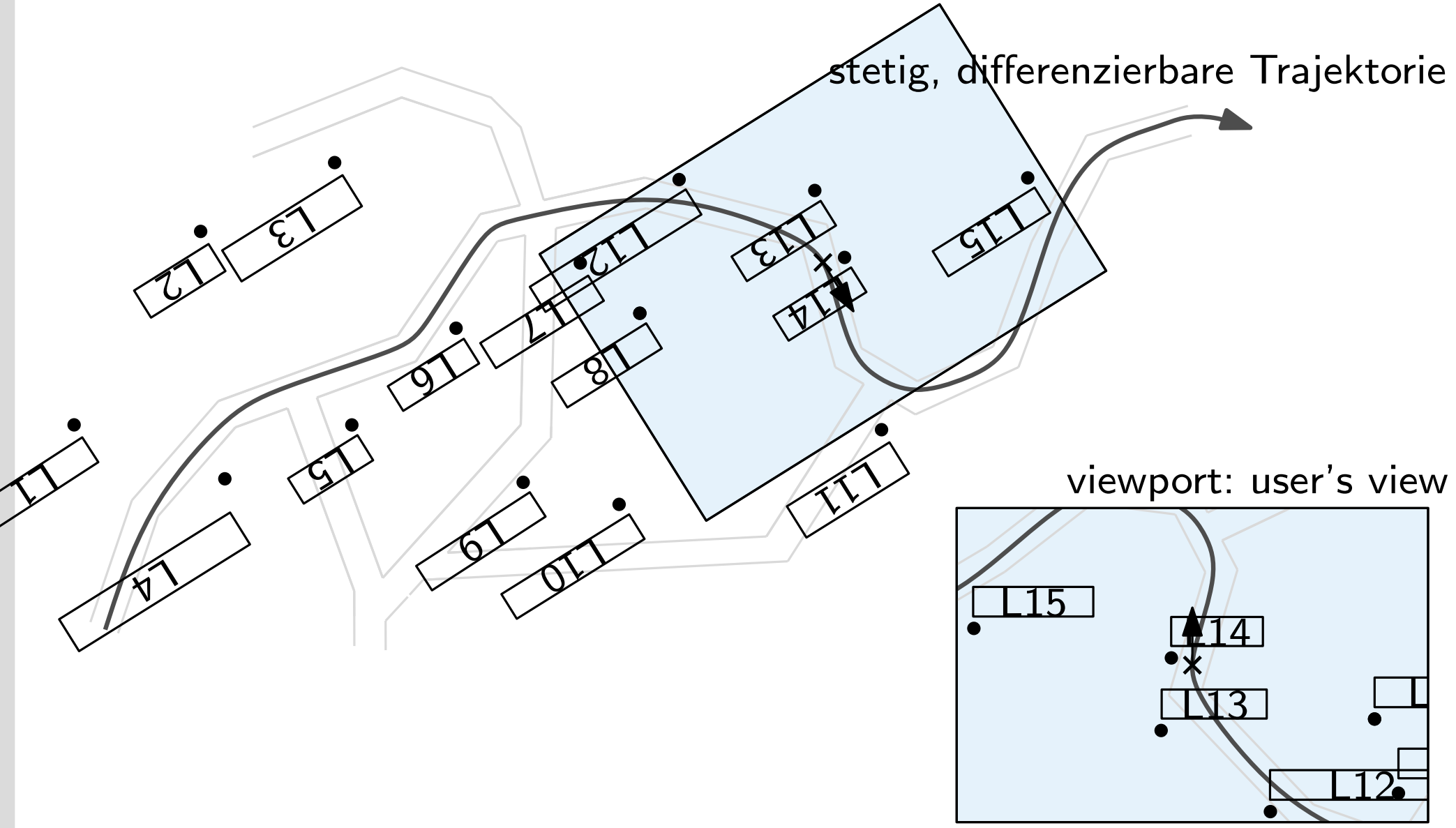
Trajectory-Based Dynamic Map Labeling



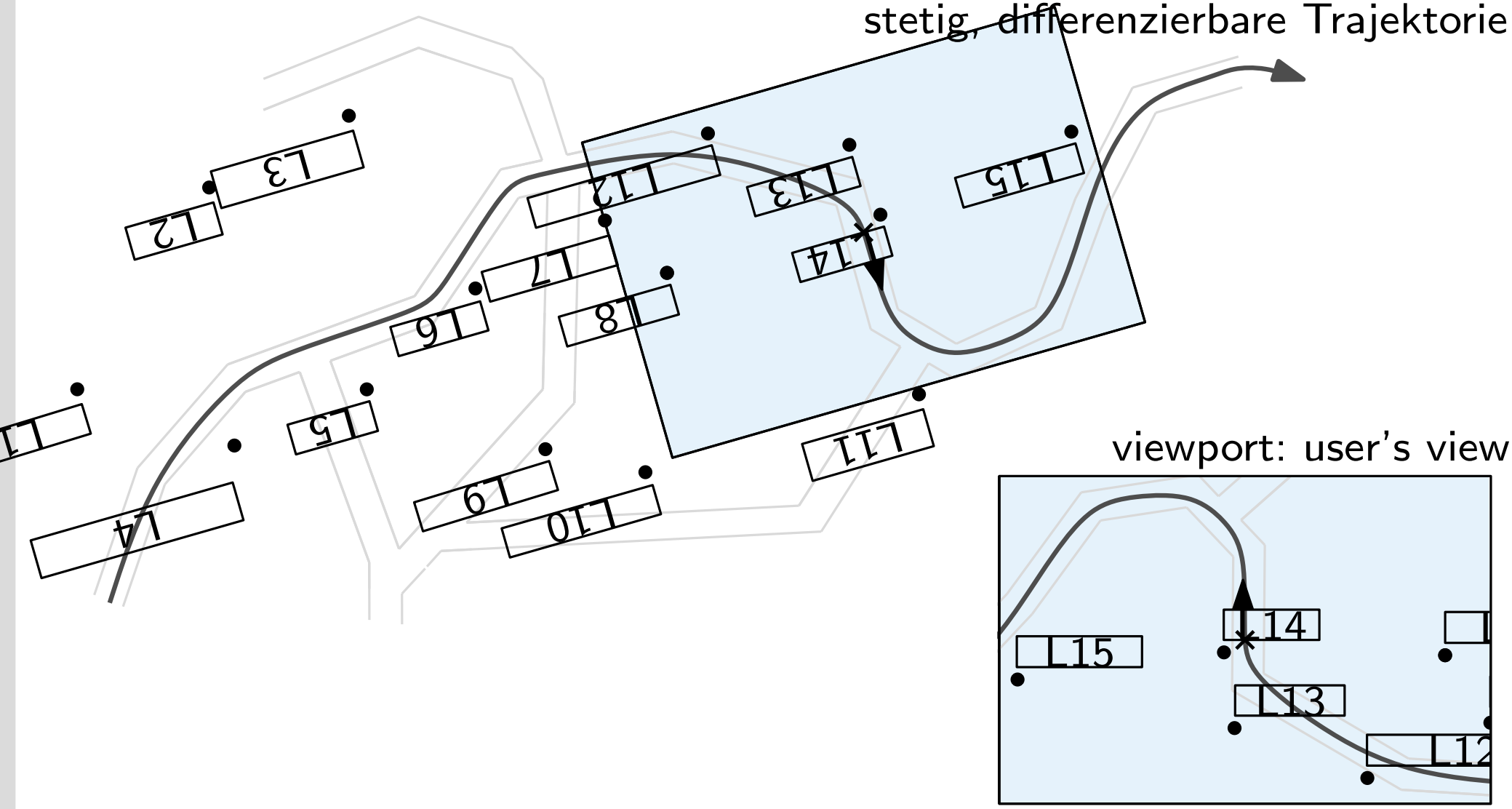
Trajectory-Based Dynamic Map Labeling



Trajectory-Based Dynamic Map Labeling



Trajectory-Based Dynamic Map Labeling



Trajectory-Based Dynamic Map Labeling

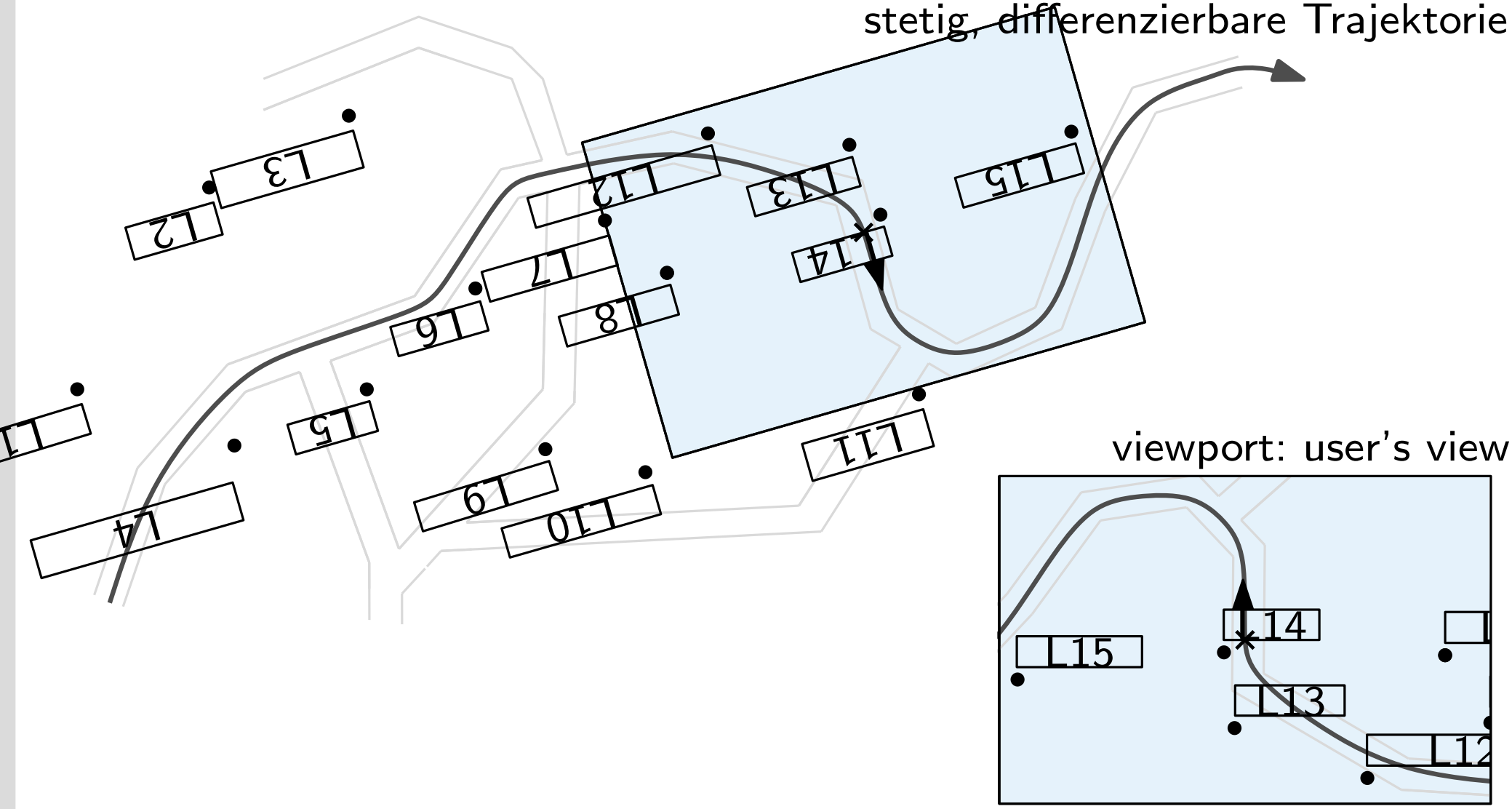
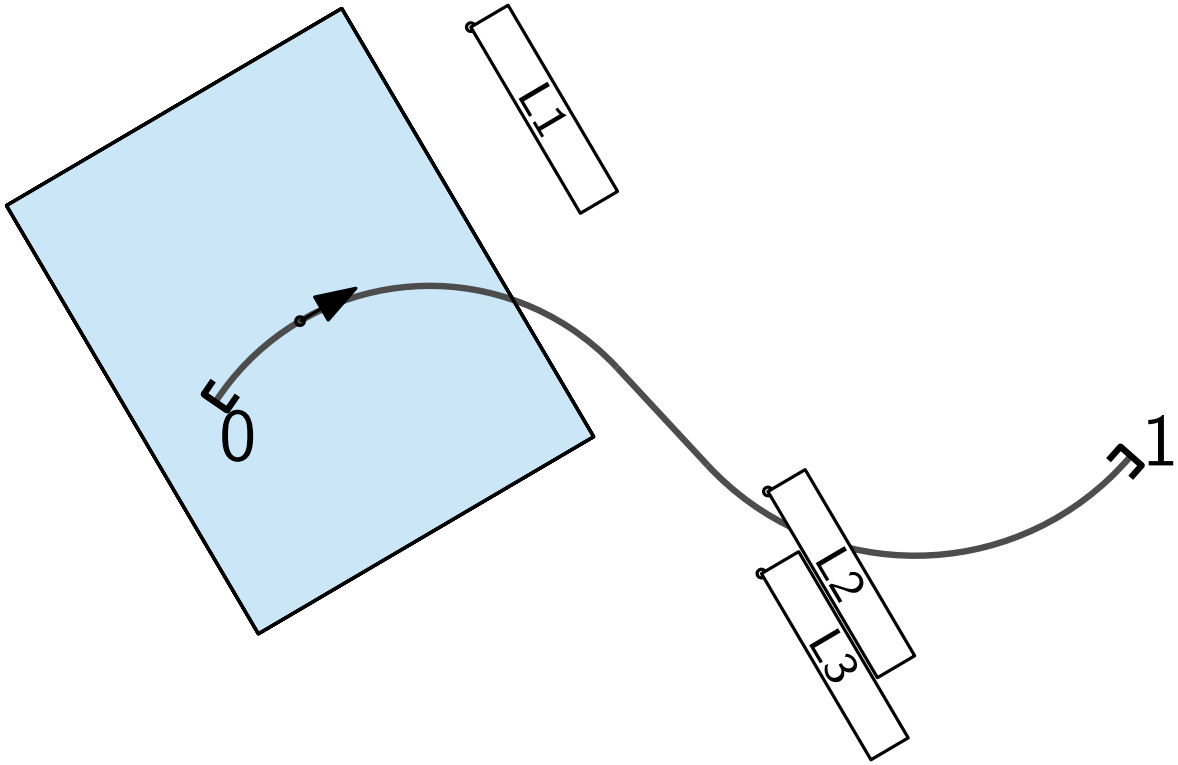


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:

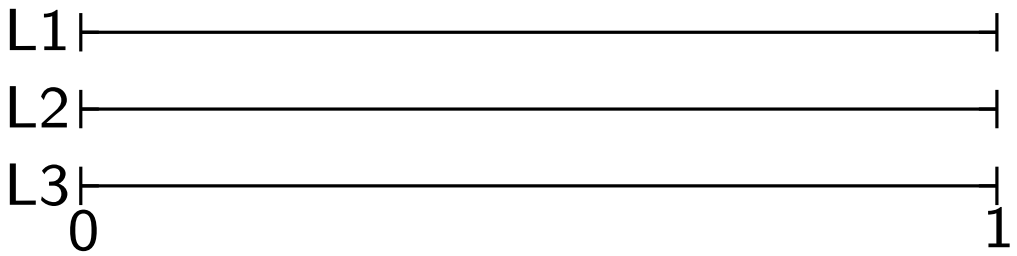
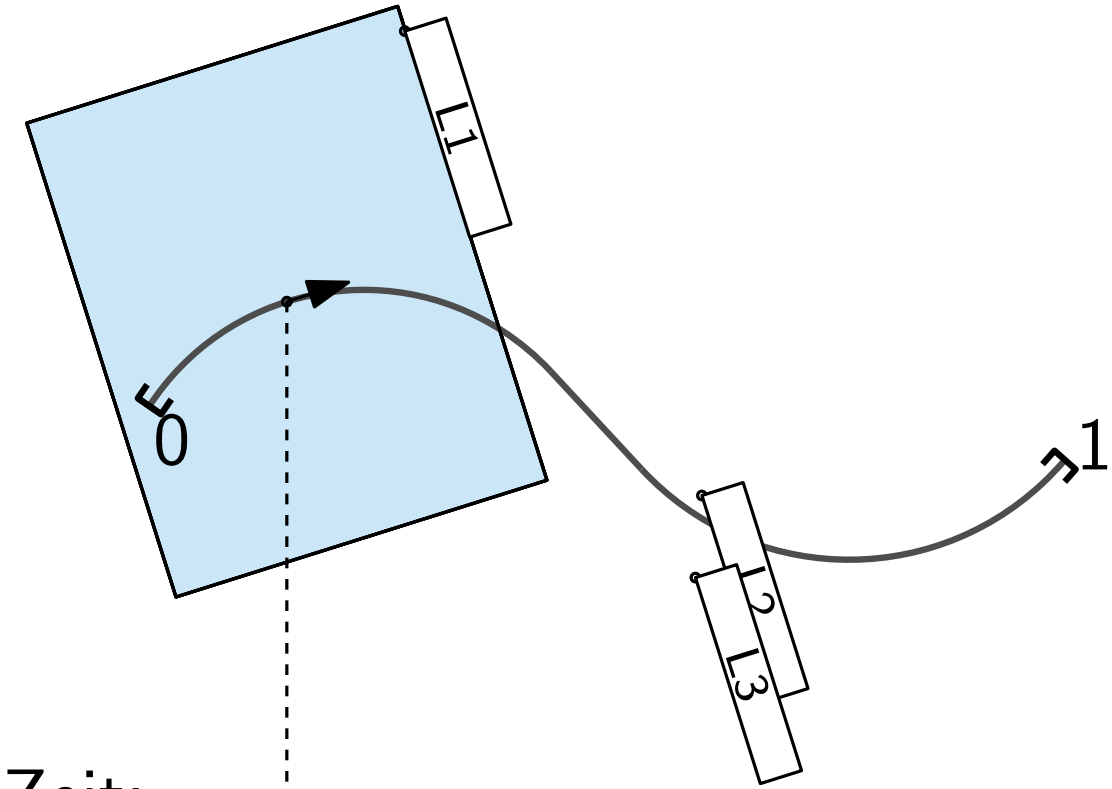


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



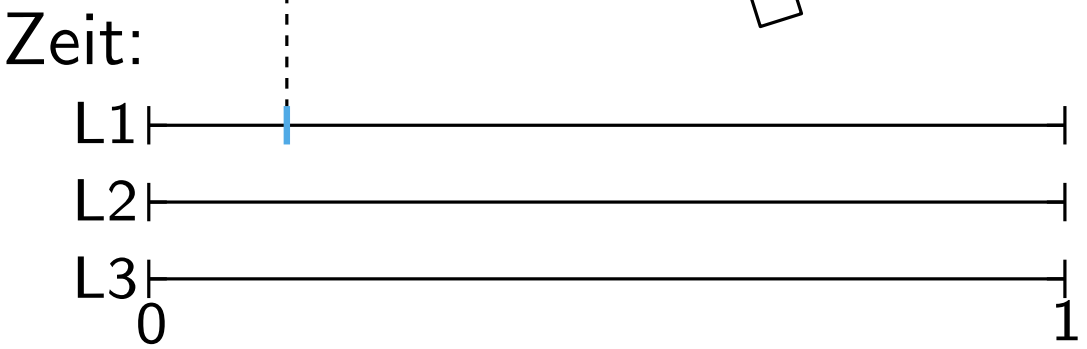
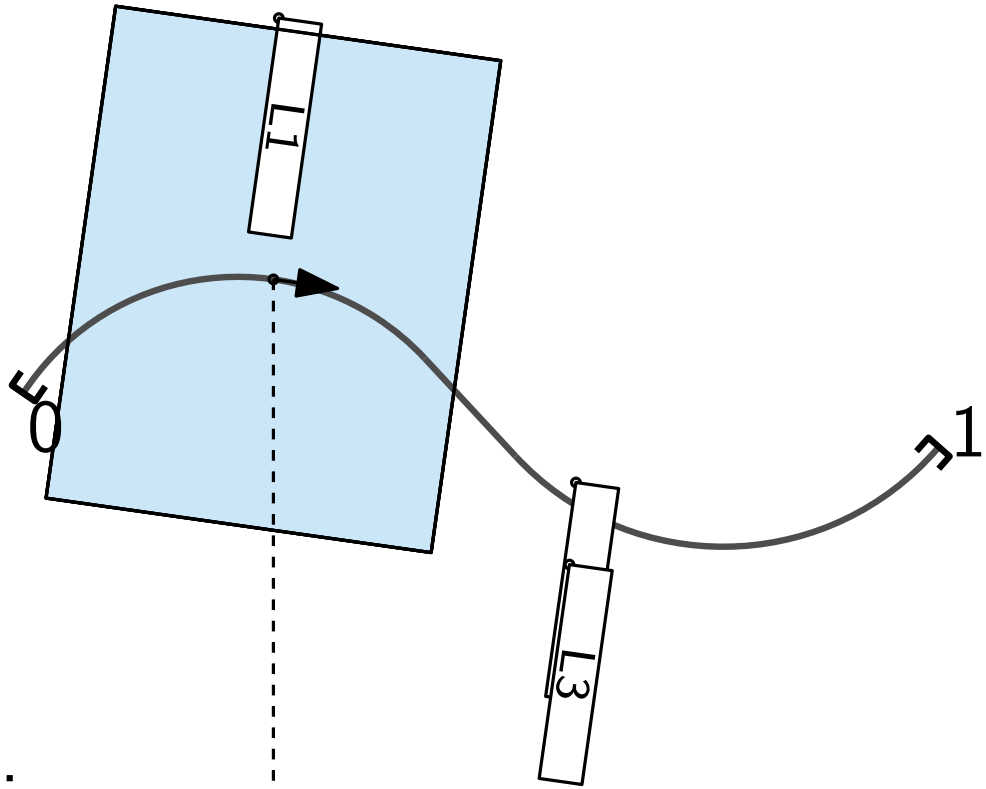


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:

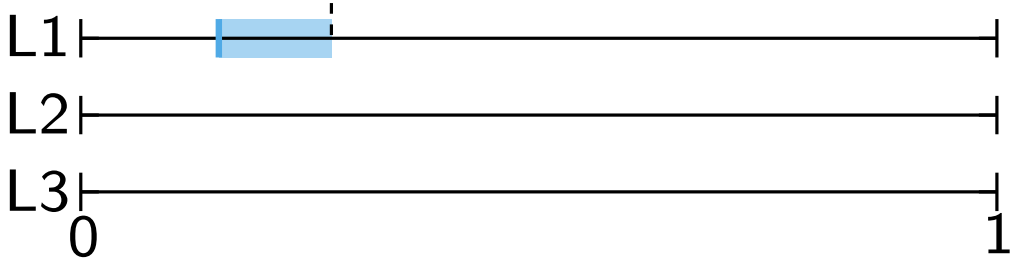
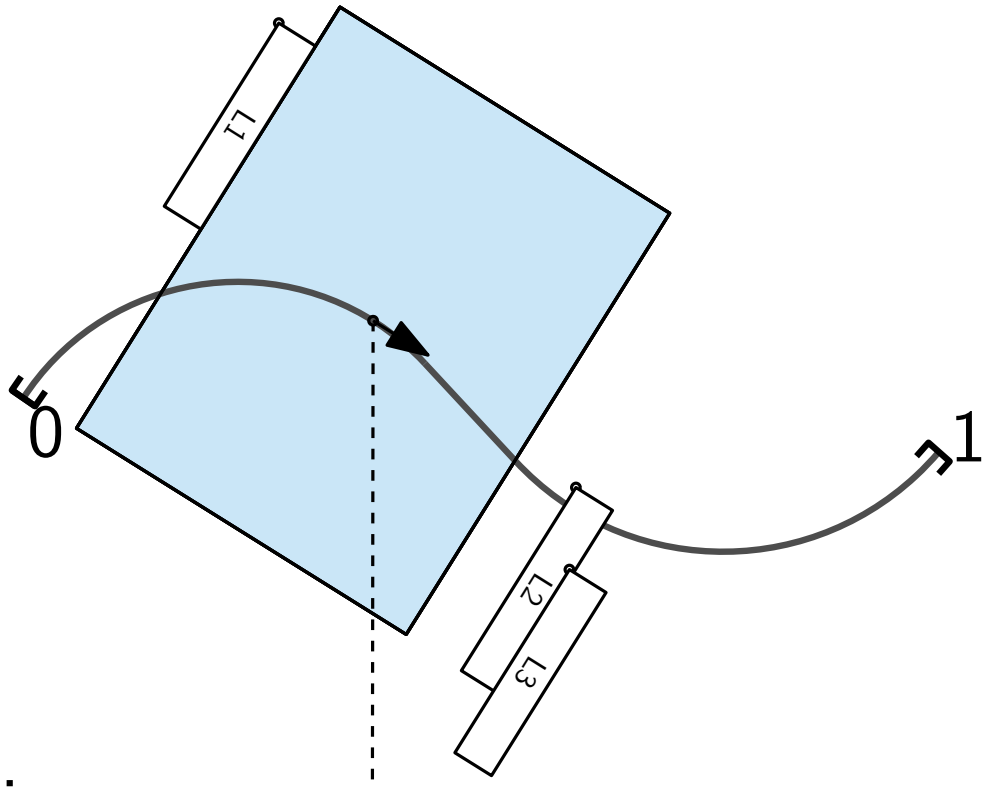


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:

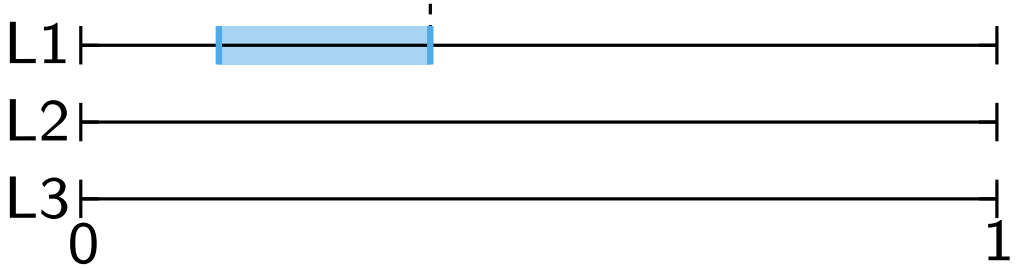
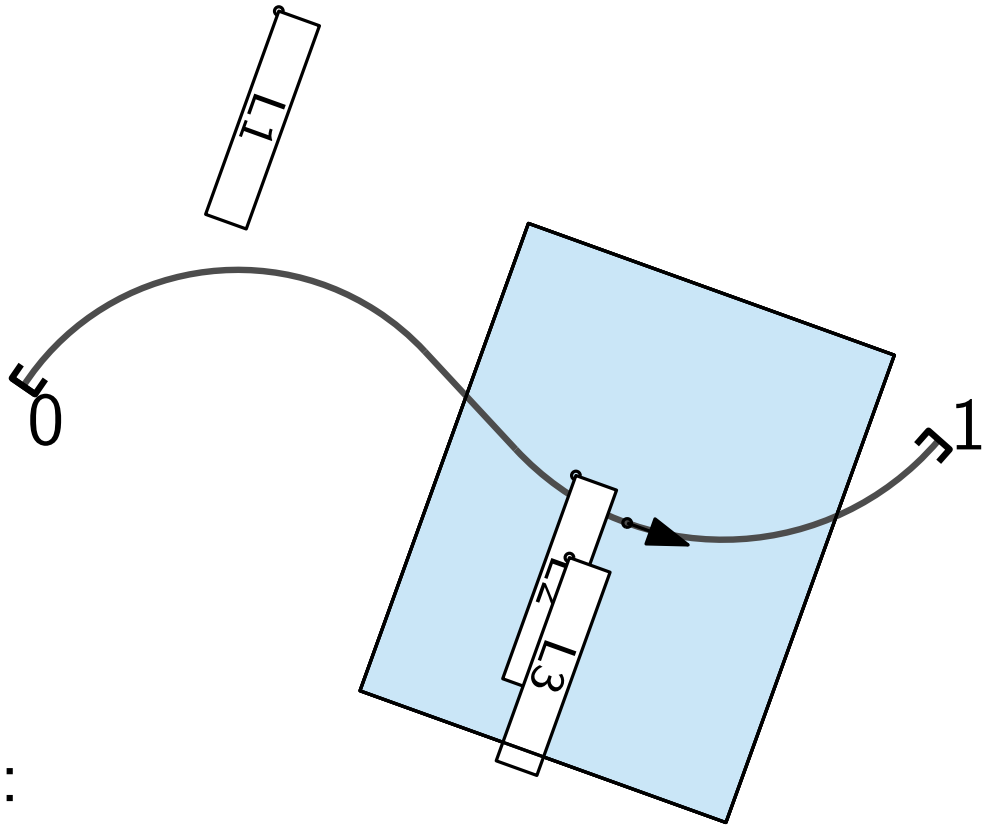
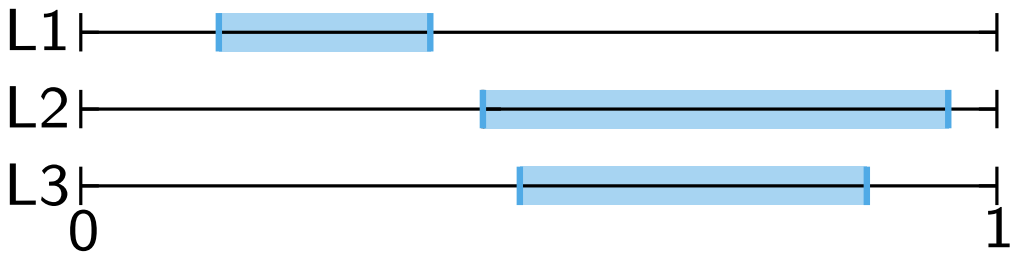


Abbildung auf Intervalle

Label ist **präsent** zum Zeitpunkt t falls es den Viewport zum Zeitpunkt t schneidet.



Zeit:



Menge von *Präsenzintervallen*

Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.

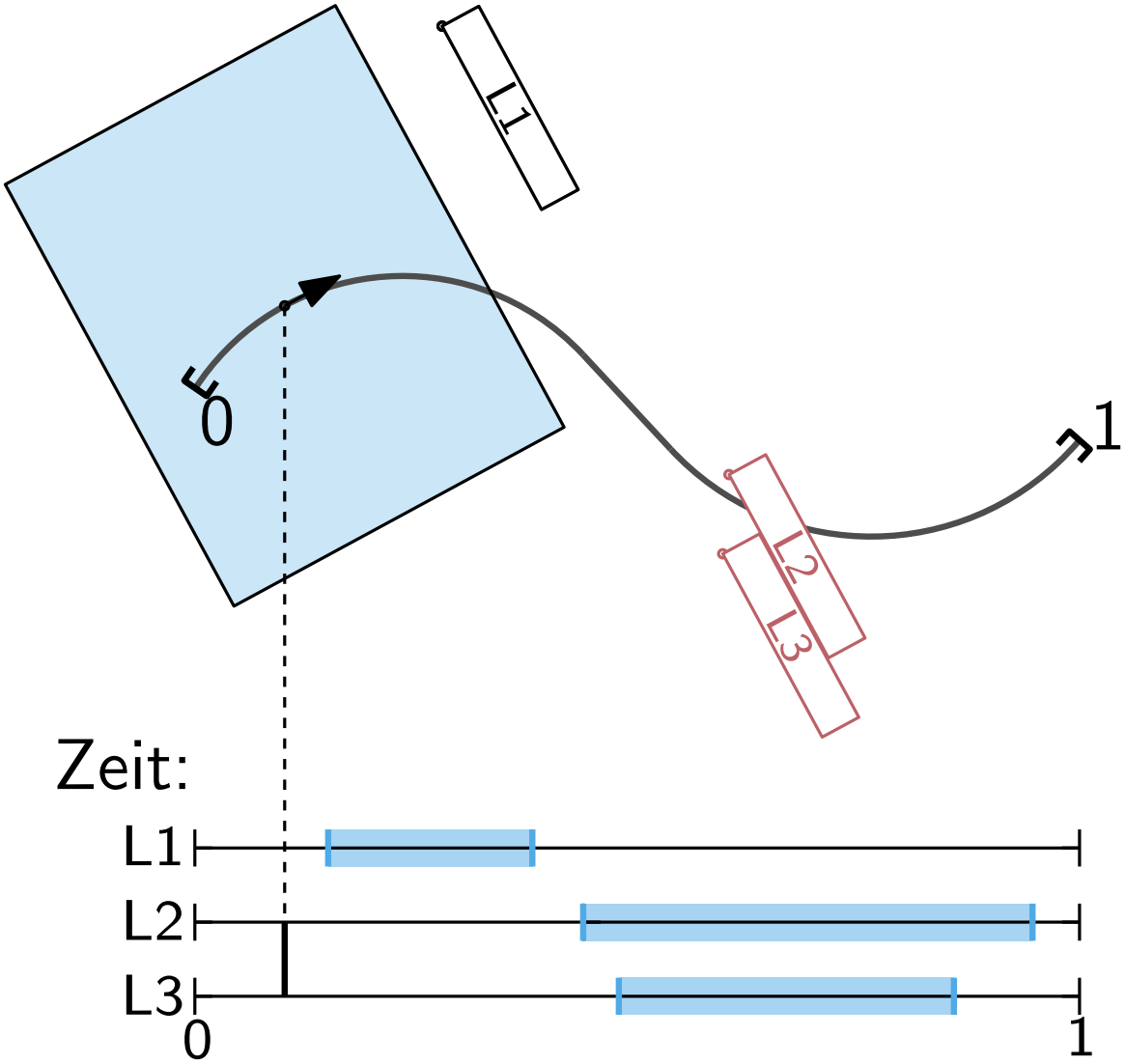
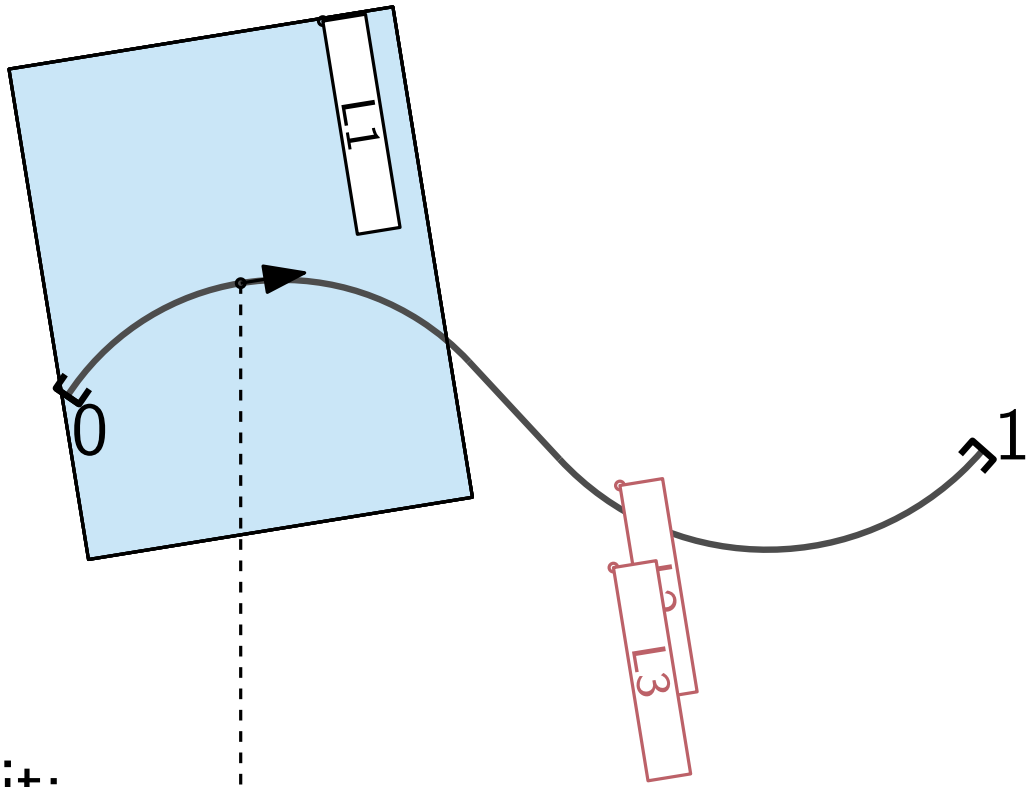


Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



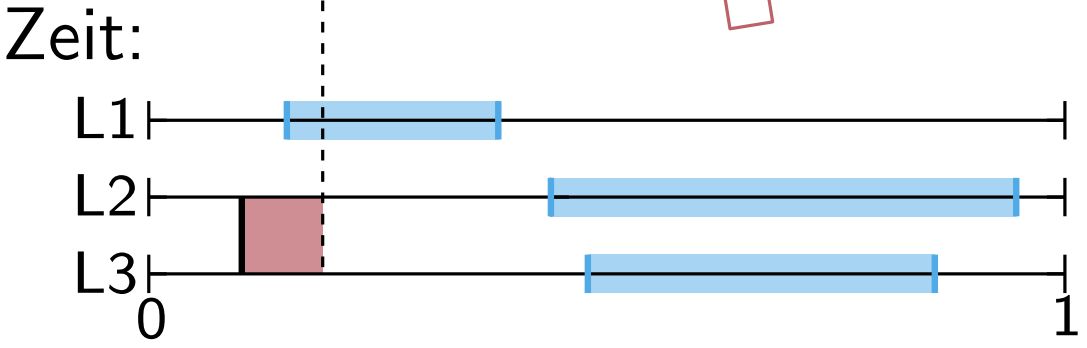
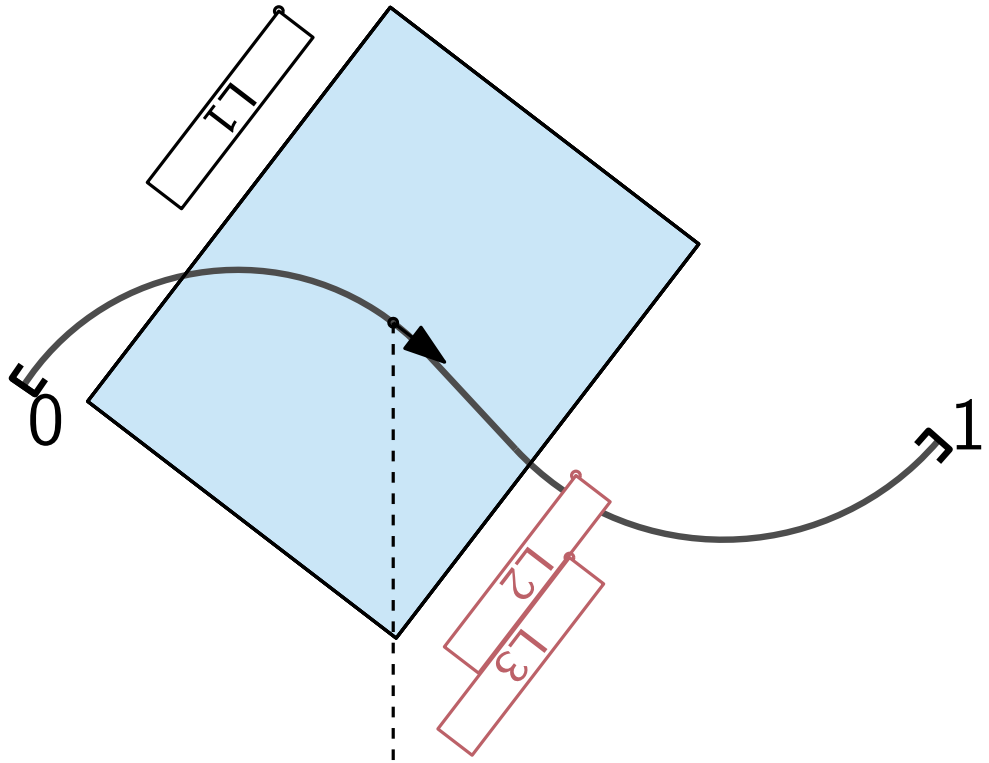


Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



Zeit:

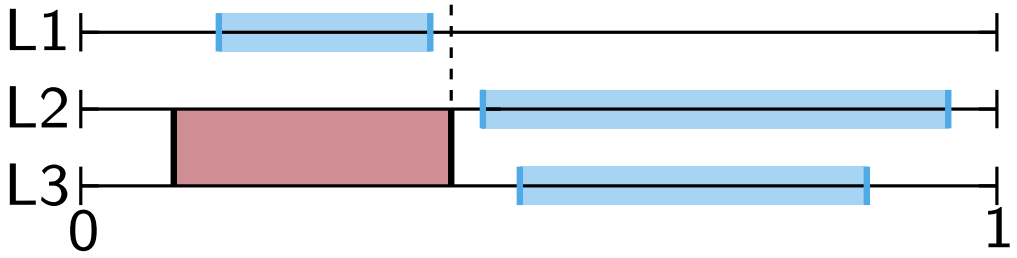
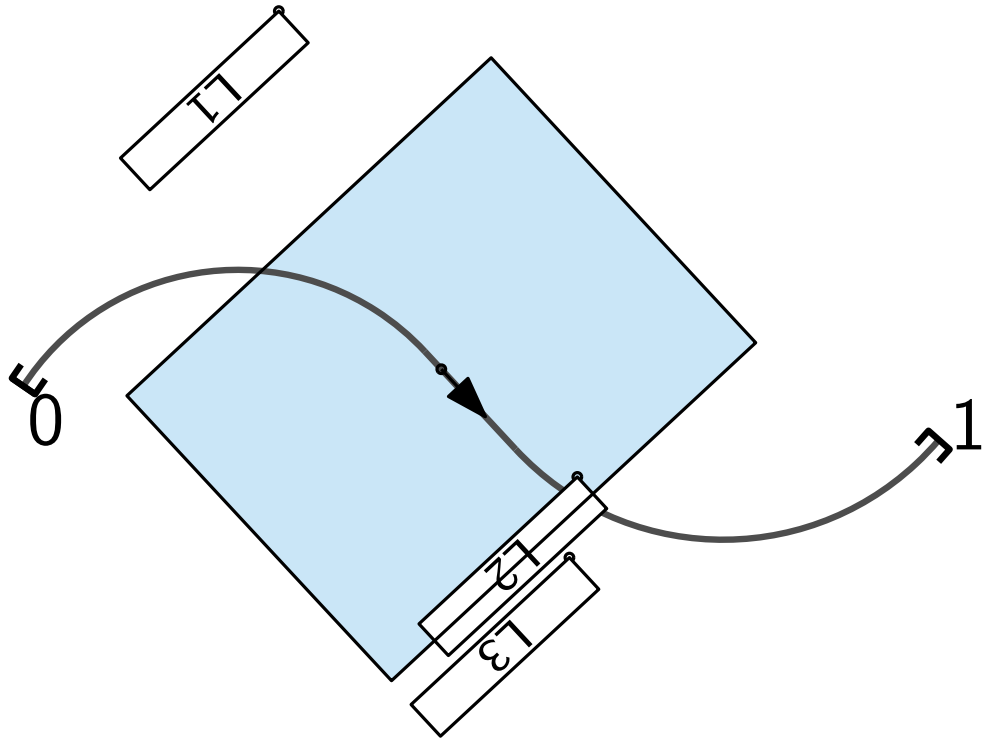


Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



Zeit:

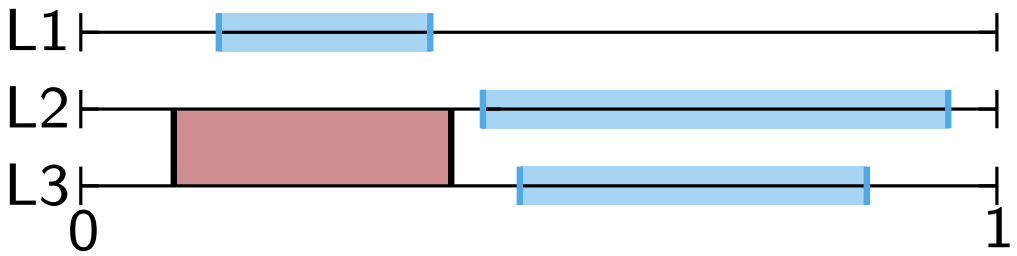
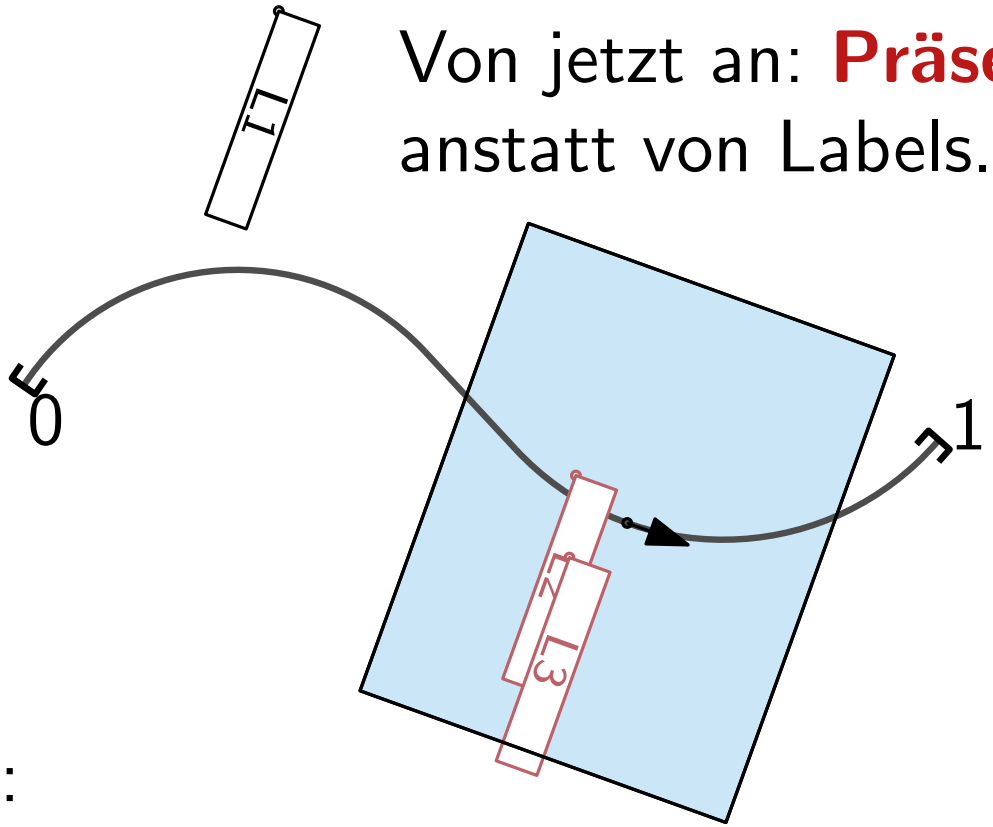


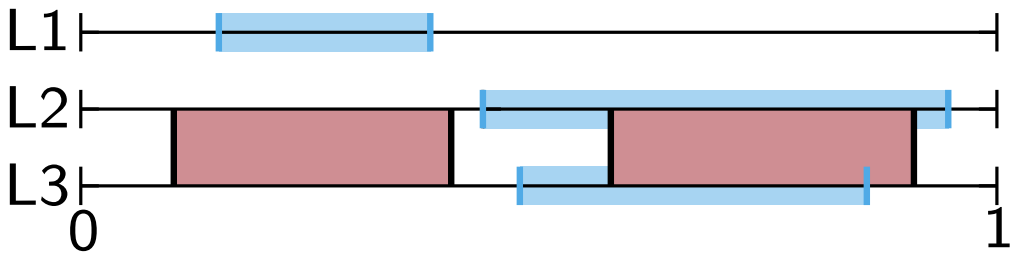
Abbildung auf Intervalle

Zwei Labels stehen in einem **Konflikt** zum Zeitpunkt t, falls sie sich zum Zeitpunkt t schneiden.



Von jetzt an: **Präsenz- und Konfliktintervalle** anstatt von Labels.

Zeit:



Menge von **Konfliktintervallen**

Problemdefinition

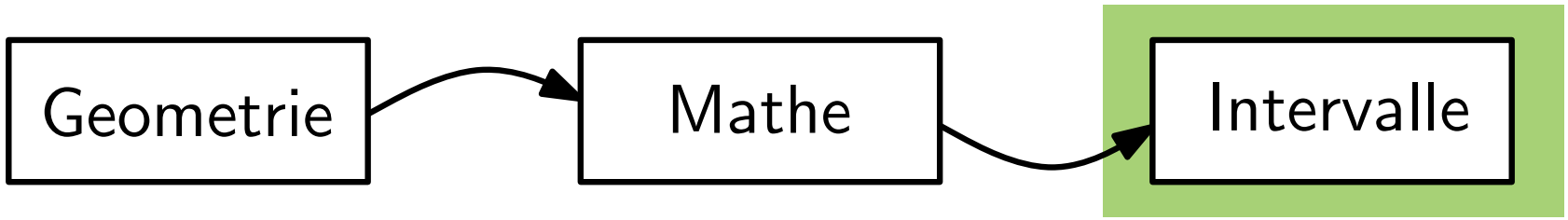
Overview

Eingabe:
Karte M, Menge P an Punkten, Menge L an Labels

Ausgabe:
Konsistente rotierende Beschriftung **die Summe aller aktiven Intervalle maximiert.**

Overview

Eingabe:
Karte M, Menge P an Punkten, Menge L an Labels

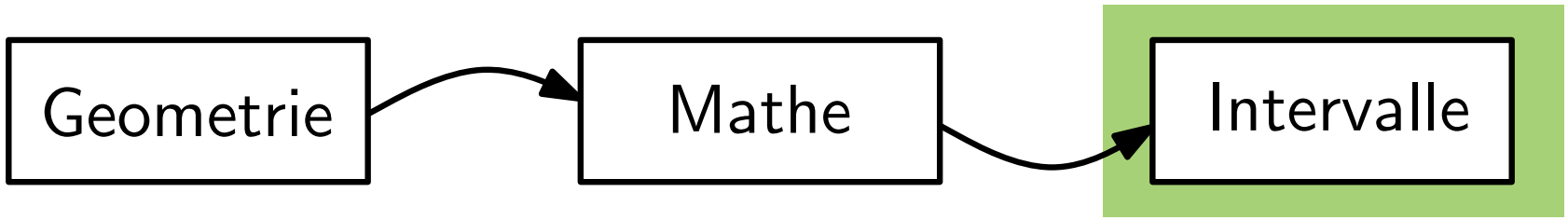


Ausgabe:
Konsistente rotierende Beschriftung **die Summe aller aktiven Intervalle maximiert.**

Overview

Eingabe:
Karte M, Menge P an Punkten, Menge L an Labels

Eingabe':
Eingabe an Präsenzintervallen, Konfliktintervallen



Ausgabe':
Menge an Aktivitätsintervallen

Ausgabe:
Konsistente rotierende Beschriftung **die Summe aller aktiven Intervalle maximiert.**

Aktivität von Labels

Label ist **aktiv** zum Zeitpkt. t falls es zu t angezeigt wird.

Aktivität von Labels

Label ist **aktiv** zum Zeitpkt. t falls es zu t angezeigt wird.

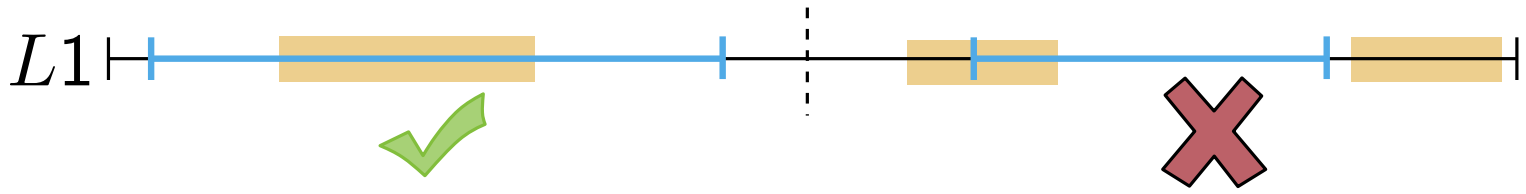
Konsistenzkriterien?

Label ist präsent. Label ist aktiv. Labels stehen im Konflikt.

Aktivität von Labels

Label ist **aktiv** zum Zeitpkt. t falls es zu t angezeigt wird.

Nur aktiv, falls auch präsent.

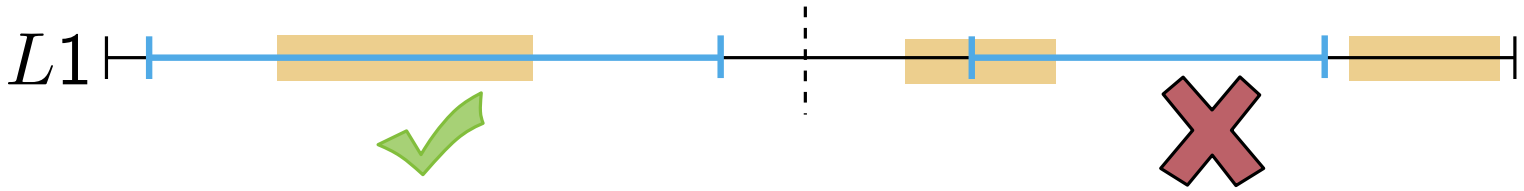


Label ist präsent. Label ist aktiv. Labels stehen im Konflikt.

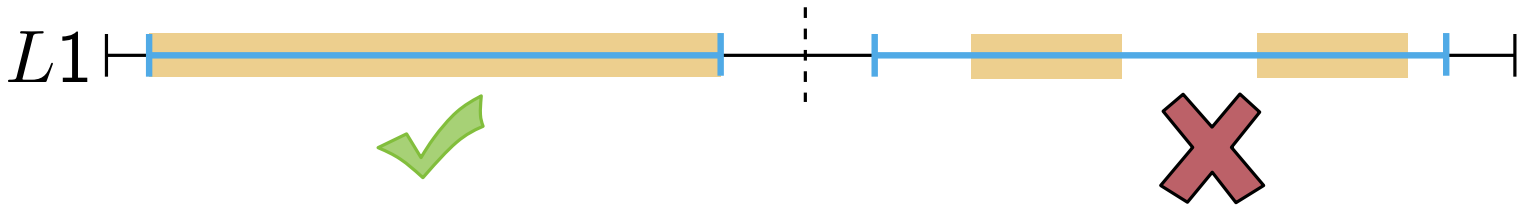
Aktivität von Labels

Label ist **aktiv** zum Zeitpkt. t falls es zu t angezeigt wird.

Nur aktiv, falls auch präsent.



Entweder aktiv für **gesamtes Präsenzintervall** oder gar nicht.

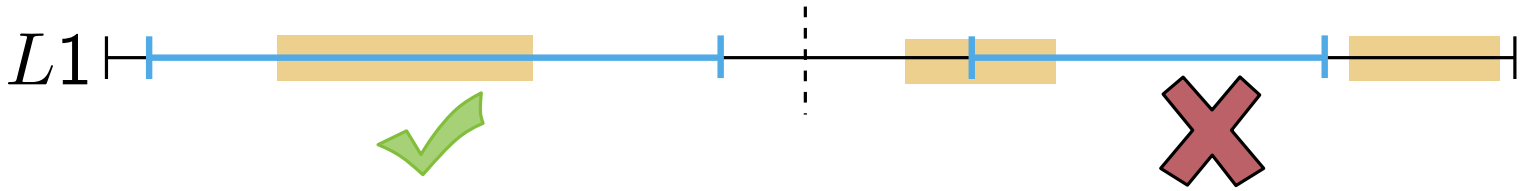


Label ist präsent. Label ist aktiv. Labels stehen im Konflikt.

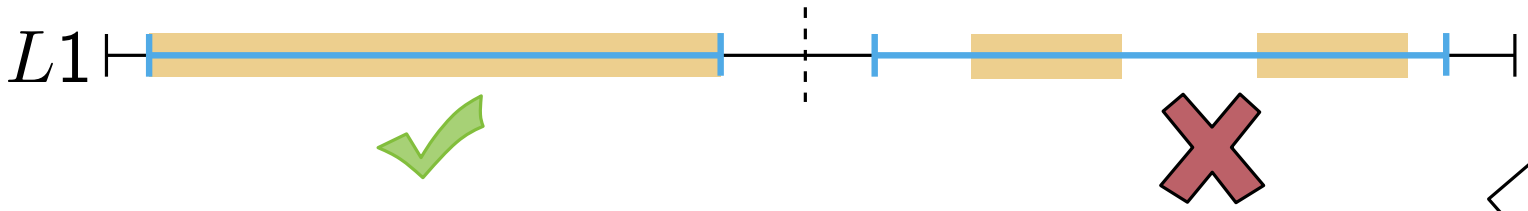
Aktivität von Labels

Label ist **aktiv** zum Zeitpkt. t falls es zu t angezeigt wird.

Nur aktiv, falls auch präsent.



Entweder aktiv für **gesamtes Präsenzintervall** oder gar nicht.



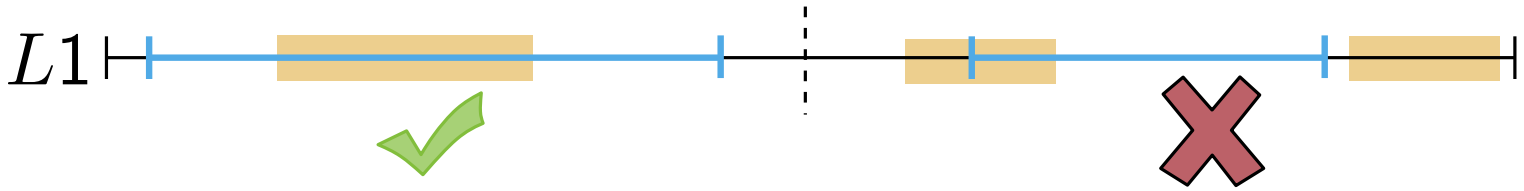
vermeidet flackern

Label ist präsent. Label ist aktiv. Labels stehen im Konflikt.

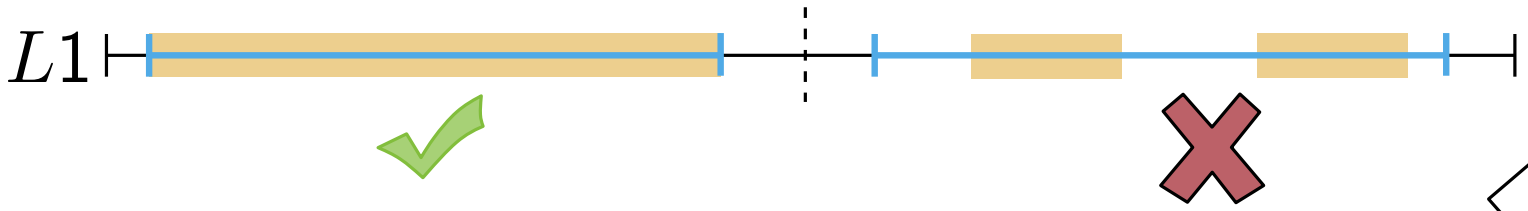
Aktivität von Labels

Label ist **aktiv** zum Zeitpkt. t falls es zu t angezeigt wird.

Nur aktiv, falls auch präsent.

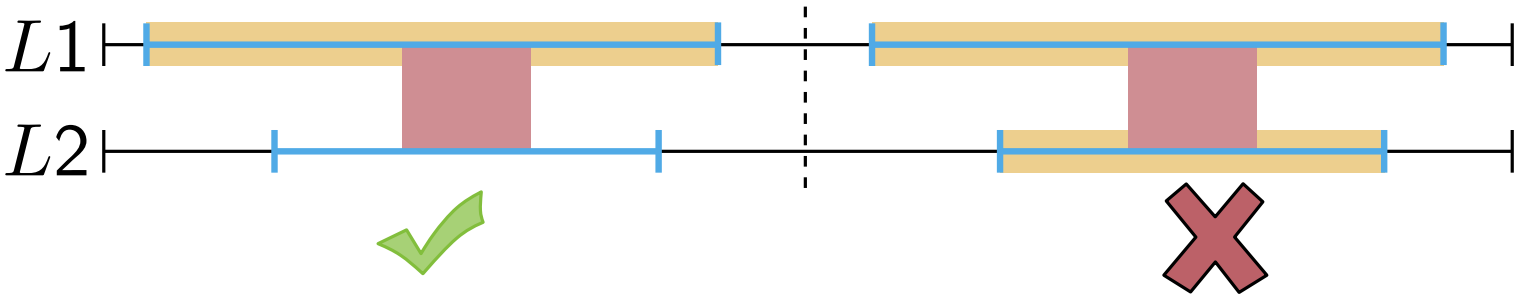


Entweder aktiv für **gesamtes Präsenzintervall** oder gar nicht.



vermeidet flackern

Keine Konflikte zwischen aktiven Labels.

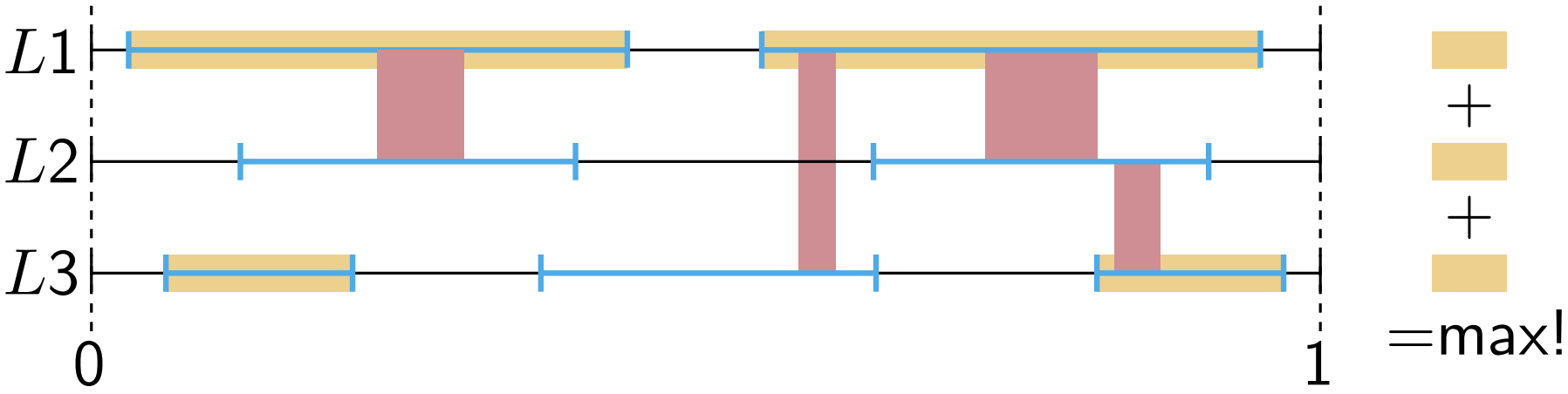


Label ist präsent. Label ist aktiv. Labels stehen im Konflikt.

Problem Definition

Problem GeneralMaxTotal:
Gegeben: Menge I an Präsenzintervallen, Konflikte
Gesucht: Teilmenge $J \subseteq I$, sodass $\sum_{j \in J} \text{length}(j)$ maximal ist, und J konfliktfrei ist.

Menge J an Intervallen ist **konfliktfrei** falls keine zwei Intervalle in J in Konflikt stehen.



Label ist präsent. Label ist aktiv. Labels stehen im Konflikt.

Problem Definition

Problem GeneralMaxTotal:
Gegeben: Menge I an Präsenzintervallen, Konflikte
Gesucht: Teilmenge $J \subseteq I$, sodass $\sum_{j \in J} \text{length}(j)$ maximal ist, und J konfliktfrei ist.

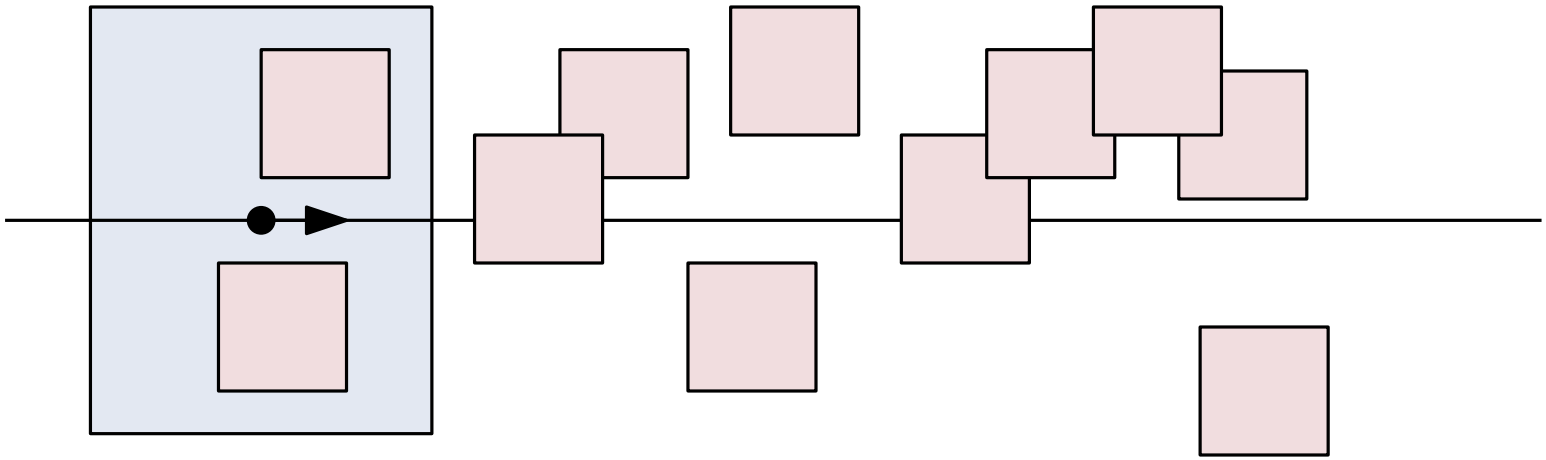
Menge J an Intervallen ist **konfliktfrei** falls keine zwei Intervalle in J in Konflikt stehen.

Theorem

GeneralMaxTotal ist NP-vollständig

Aufgabe: Beweis.

Geradlinige Kamerafahrt

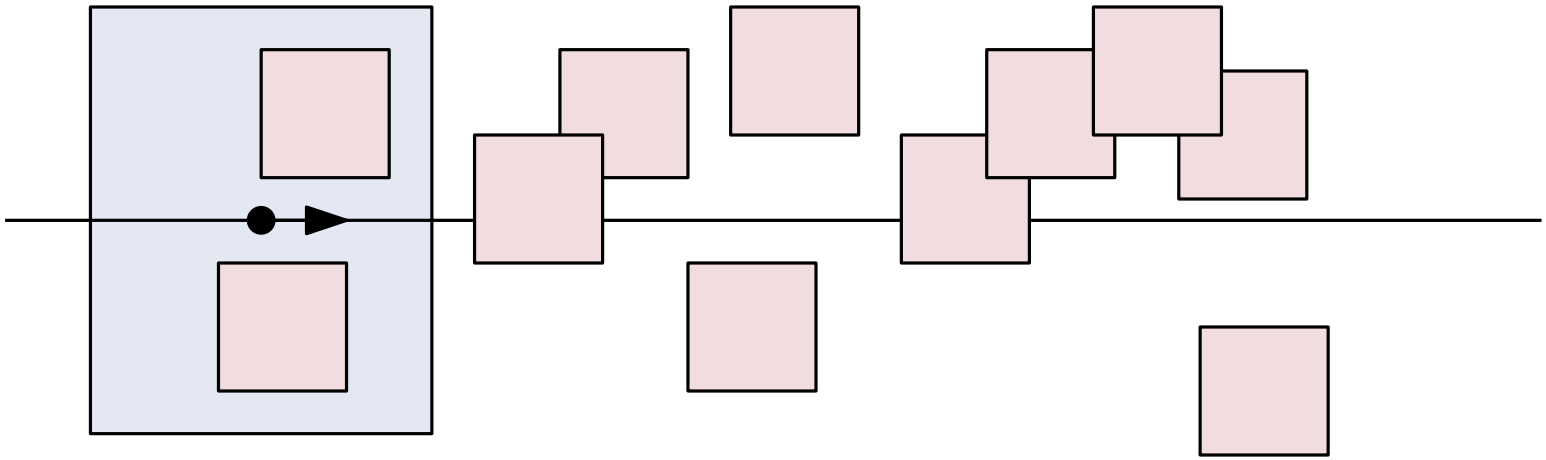


Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Gesucht: $\frac{1}{2}$-Approximation

Geradlinige Kamerafahrt



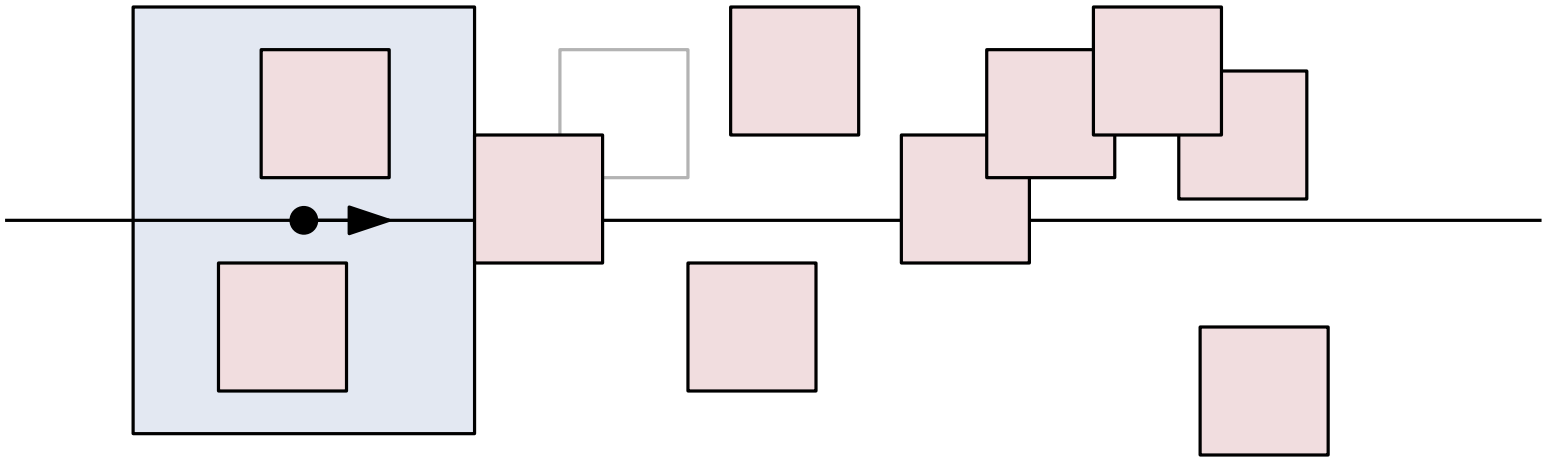
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



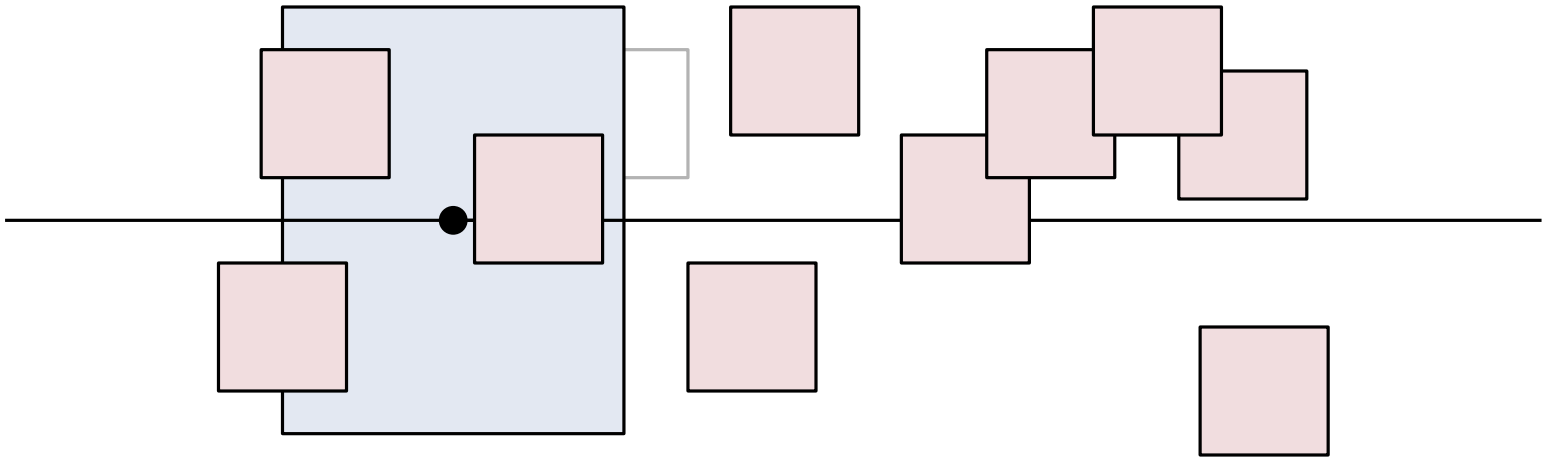
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



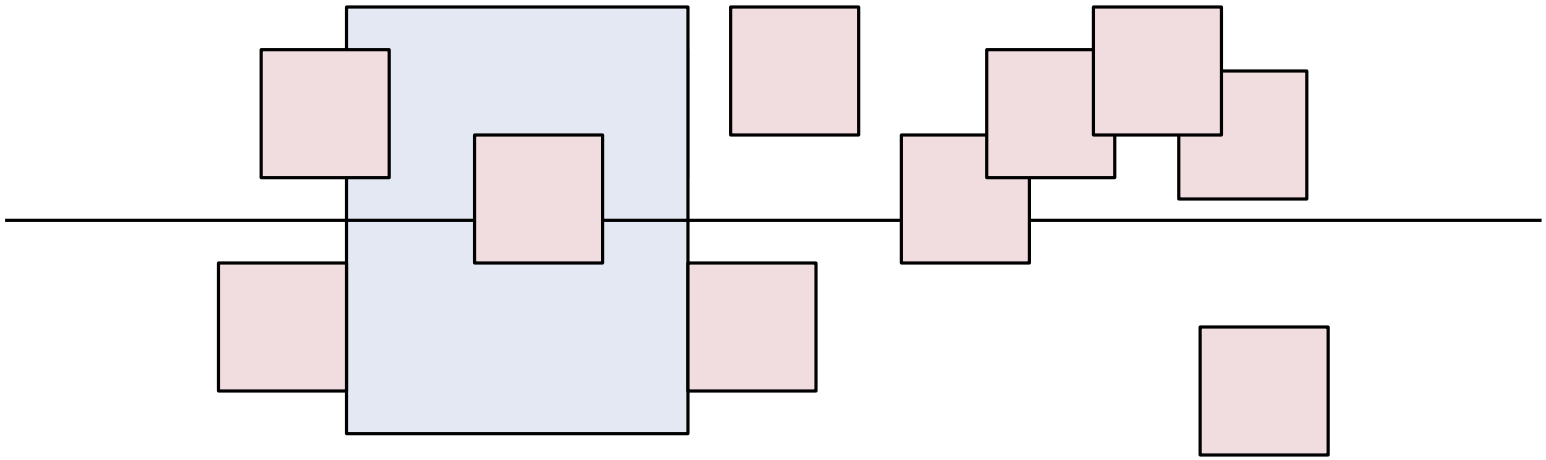
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



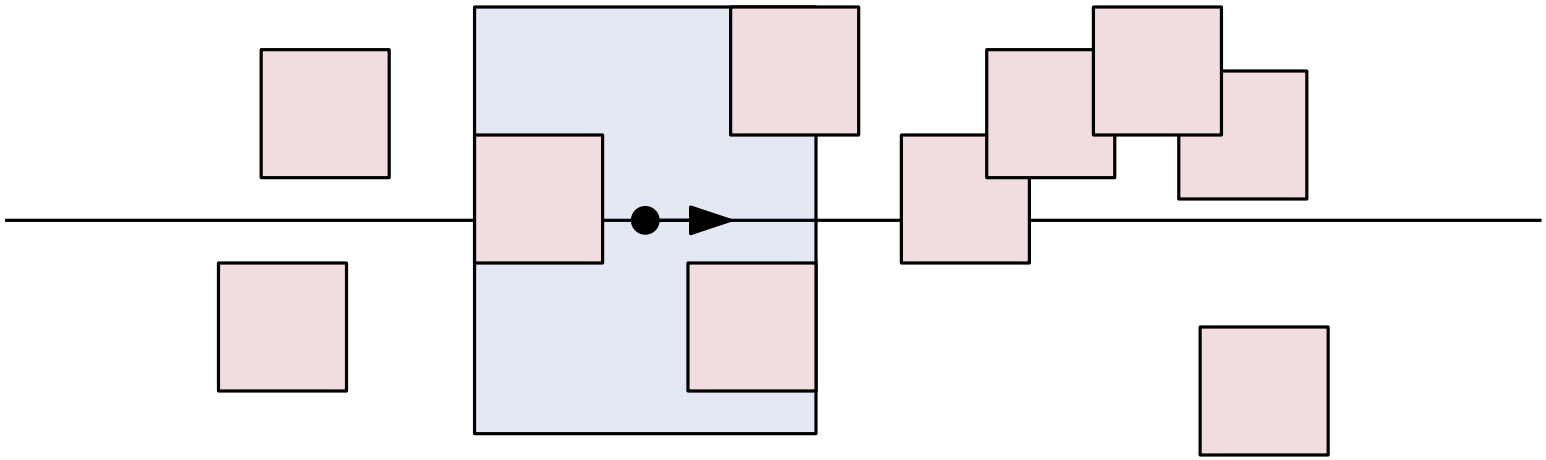
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

> Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



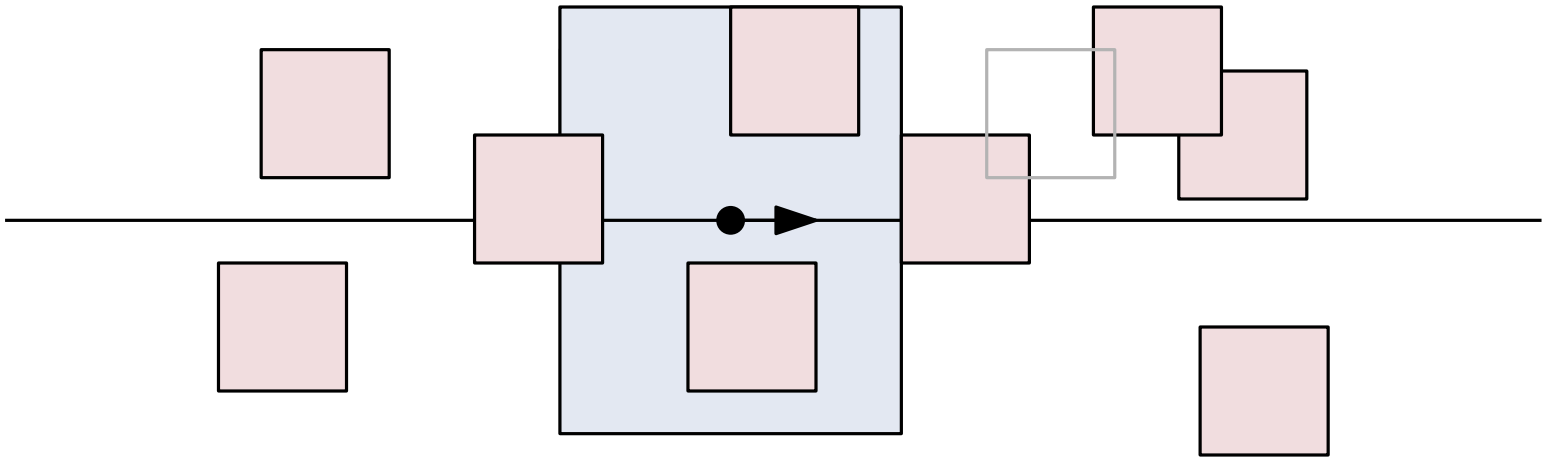
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



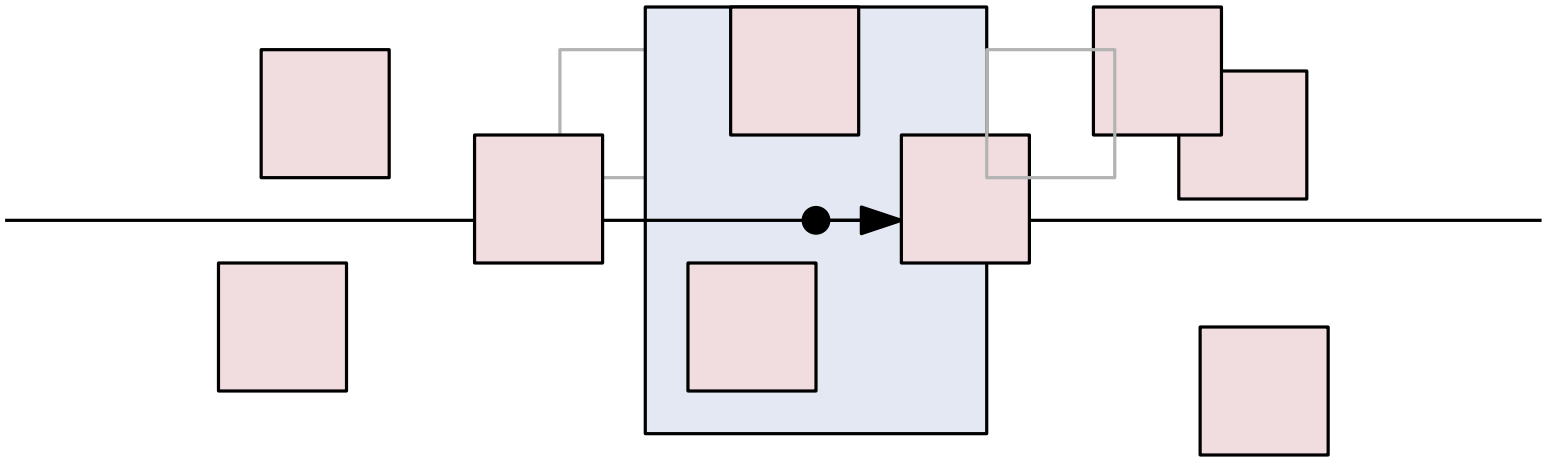
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



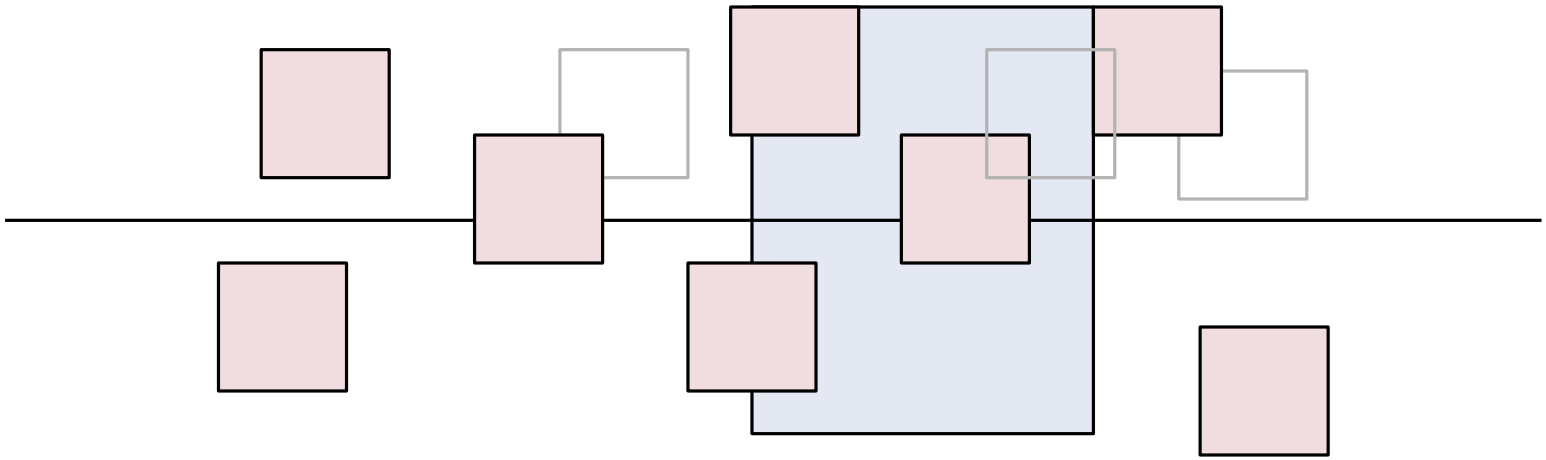
Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt



Annahme:

- Labels sind Quadrate (und in allgemeiner Lage).
- Trajektorie ist horizontale Gerade (von links nach rechts).

Greedy-Verfahren:

Wenn zwei Labels im Konflikt stehen, dann schalte dasjenige von beiden aktiv, das zuerst in den Viewport eintritt und das andere inaktiv.

Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung

$\mathcal{L}'$: Greedy-Lösung

Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung

$\mathcal{L}'$: Greedy-Lösung

1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

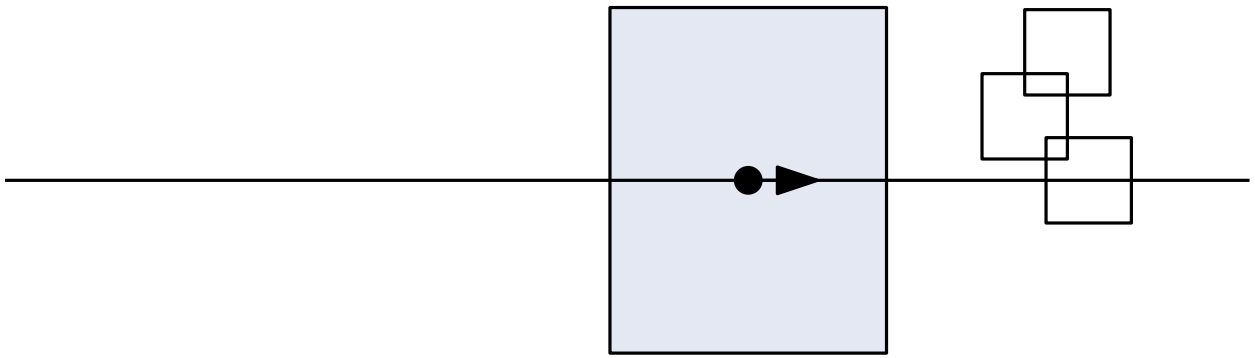
Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung $\mathcal{L}'$: Greedy-Lösung

1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

2. Weise jedes Label $\ell \in \mathcal{L} \setminus \mathcal{L}'$ dem Label $\ell' \in \mathcal{L}'$ zu, das am weitesten links liegt und ℓ schneidet:



Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

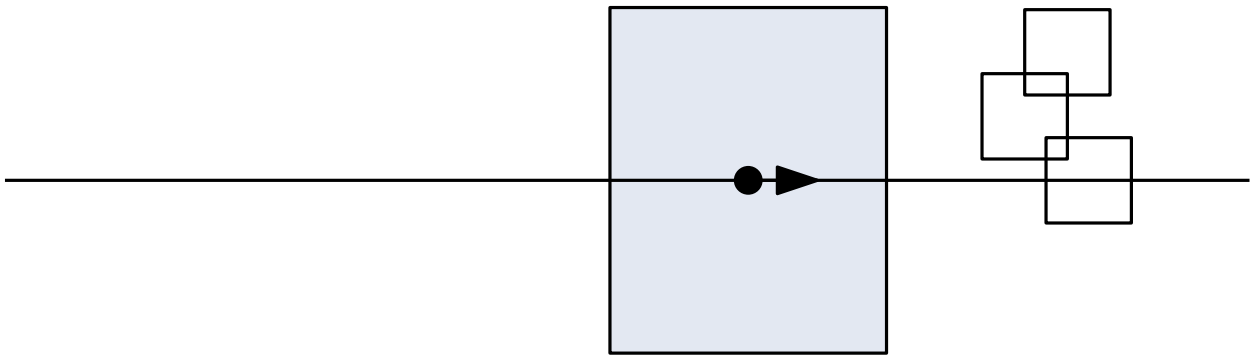
$\mathcal{L}$: optimale Lösung $\mathcal{L}'$: Greedy-Lösung

1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

2. Weise jedes Label $\ell \in \mathcal{L} \setminus \mathcal{L}'$ dem Label $\ell' \in \mathcal{L}'$ zu, das am weitesten links liegt und ℓ schneidet:

Wahl von ℓ': Mittelpunkt von ℓ' liegt weiter links, als der von ℓ.

+Einheitsquadrate: ℓ überdeckt mindestens eine rechte Ecke von ℓ', aber keine linke Ecke von ℓ.



Geradlinige Kamerafahrt

Warum ist das vorgeschlagene Verfahren eine $\frac{1}{2}$-Approximation?

$\mathcal{L}$: optimale Lösung $\mathcal{L}'$: Greedy-Lösung

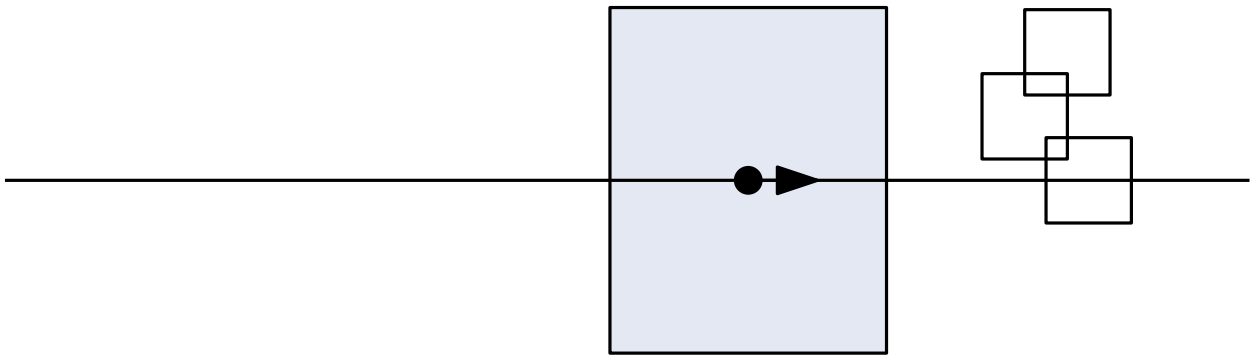
1. Weise jedes Label $\ell \in \mathcal{L} \cap \mathcal{L}'$ sich selbst zu.

2. Weise jedes Label $\ell \in \mathcal{L} \setminus \mathcal{L}'$ dem Label $\ell' \in \mathcal{L}'$ zu, das am weitesten links liegt und ℓ schneidet:

Wahl von ℓ': Mittelpunkt von ℓ' liegt weiter links, als der von ℓ.

+Einheitsquadrate: ℓ überdeckt mindestens eine rechte Ecke von ℓ', aber keine linke Ecke von ℓ.

→ Maximal zwei Labels der optimalen Lösung können ℓ' überlappen.



Ankündigung

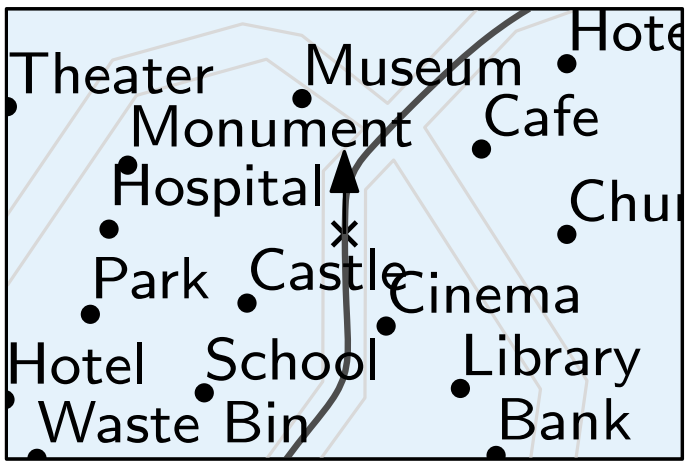
Nächste Woche Dienstag findet Evaluation statt:

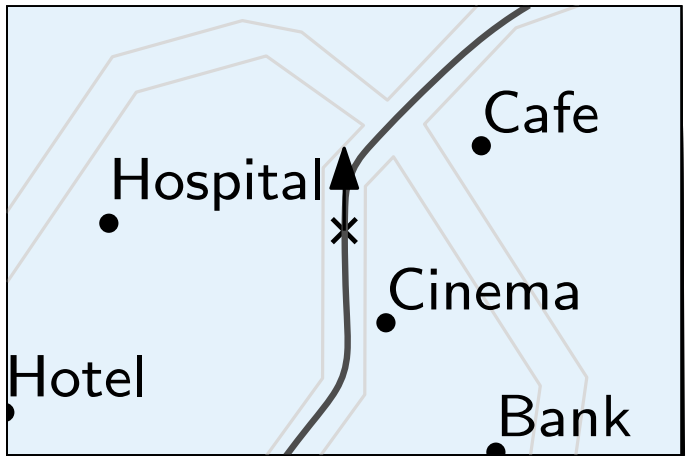
- Studenten der PH Karlsruhe evaluieren im Rahmen eines Seminars die Vorlesung.
- In den letzten 15-20min der Vorlesung Dikussionsrunde (ohne Dozent).
- Zusätzlich zur üblichen Evaluation.
- Hilft die Lehrveranstaltung zu verbessern/auf Wünsche/Belange der Teilnehmer einzugehen.

Problemdefinition

Problem k-RestrictedMaxTotal:
Gegeben: Menge I an Präsenzintervallen, Konflikte
Gesucht: $J \subseteq I$ sodass $\sum_{j \in J} \text{length}(j)$ maximal und J konfliktfrei ist, und sodass **es keinen Zeitpunkt t gibt, der von mehr als k Intervallen in J enthalten ist**.

Menge J an Intervallen ist **konfliktfrei** falls keine zwei Intervalle in J im Konflikt stehen.





$k = 5$

k-RestrictedMaxTotal

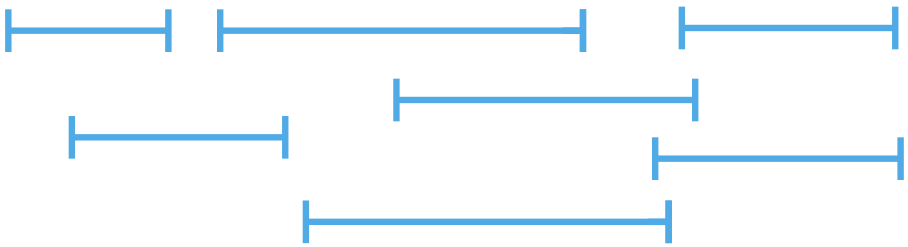
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

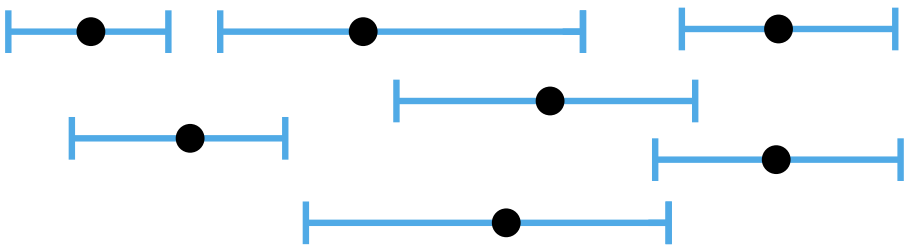
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

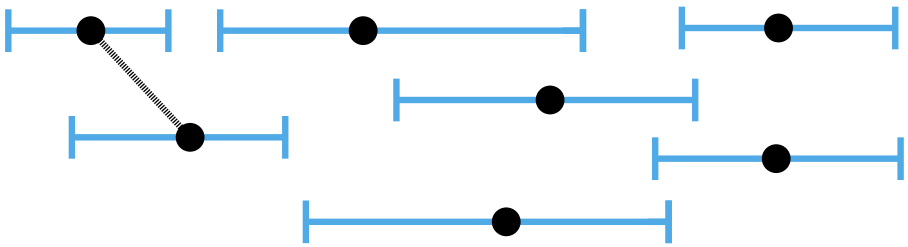
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

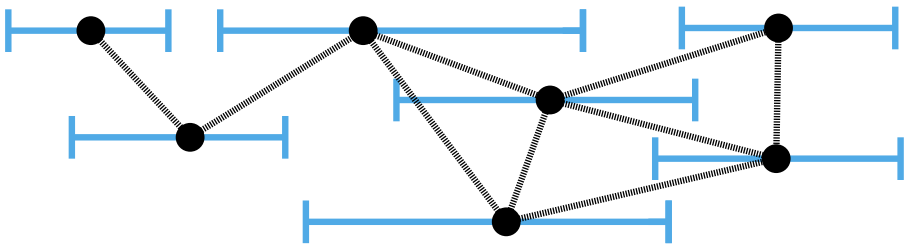
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



k-RestrictedMaxTotal

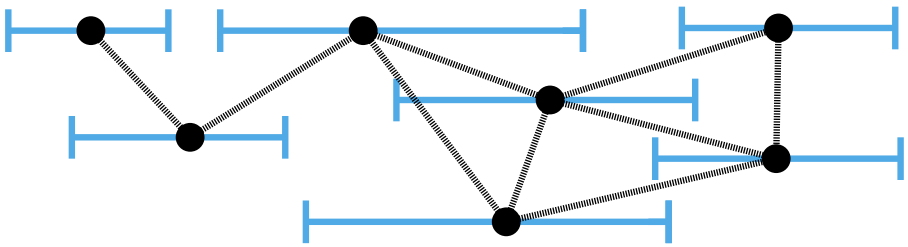
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



Gewicht eines Knoten = Länge eines Intervals

k-RestrictedMaxTotal

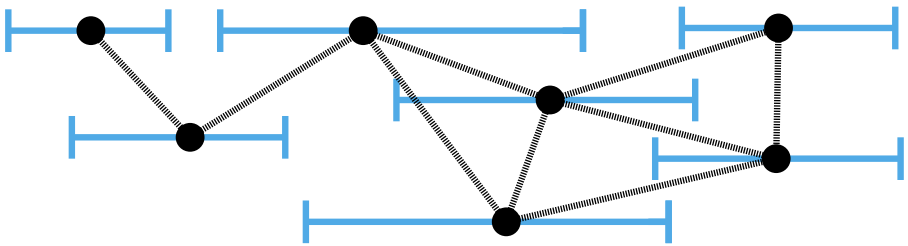
Theorem 4

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

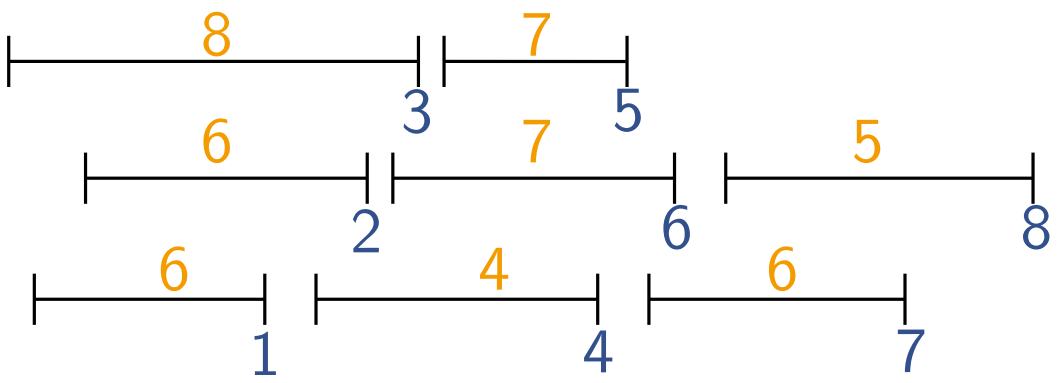
[Hsiao et al., 1992]



Gewicht eines Knoten = Länge eines Intervals

s. Übungsblatt

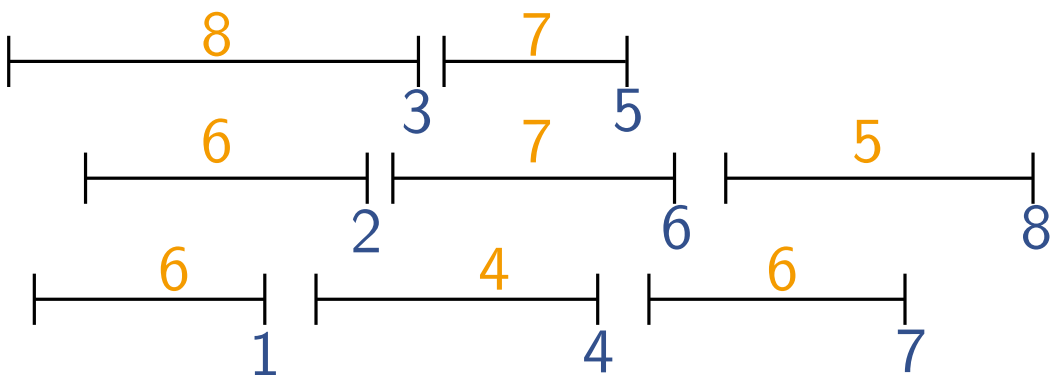
Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Lösung

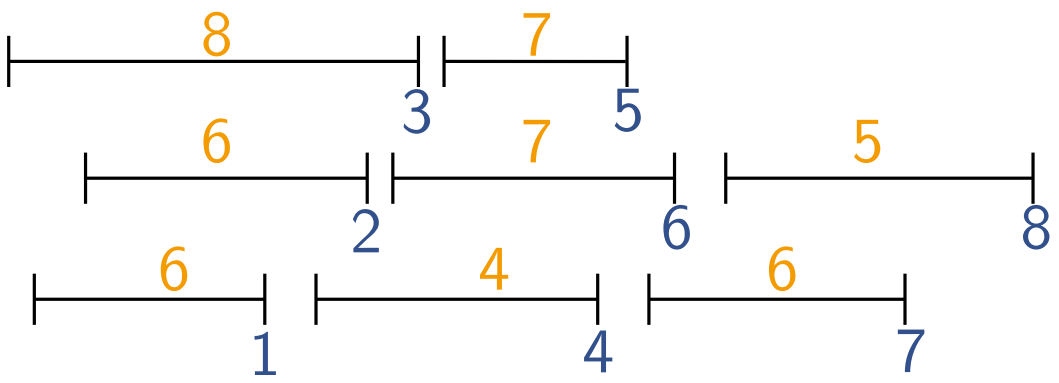


Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

Lösung



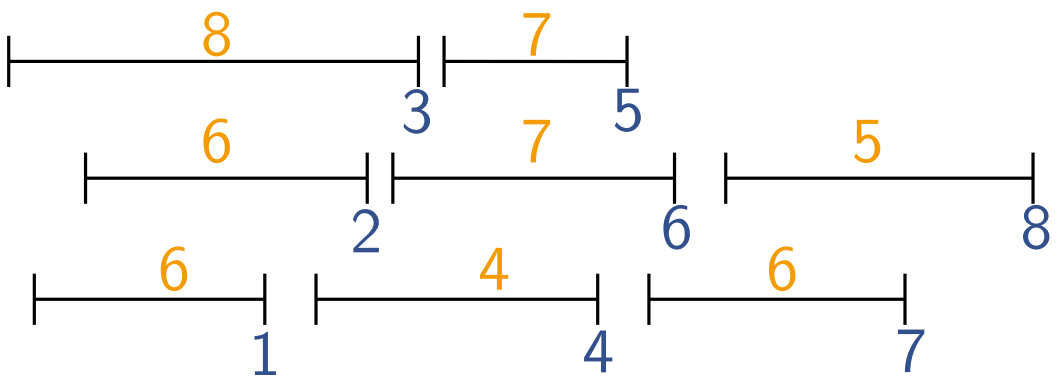
Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

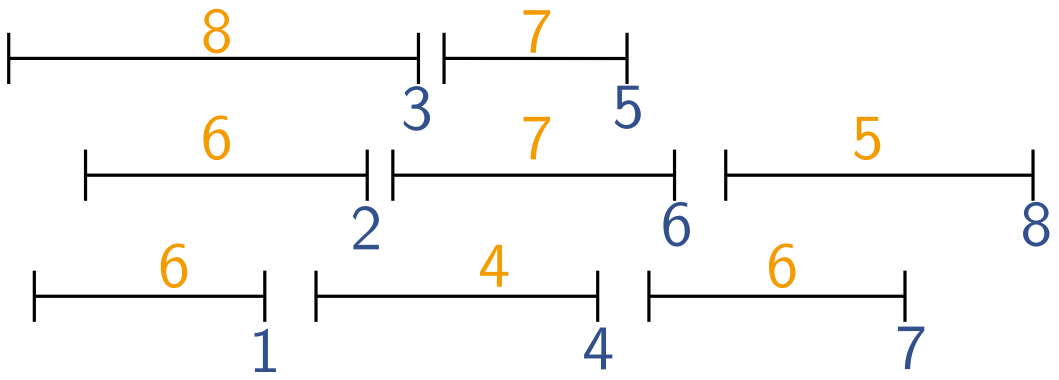
Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

$T[0] := 0$

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

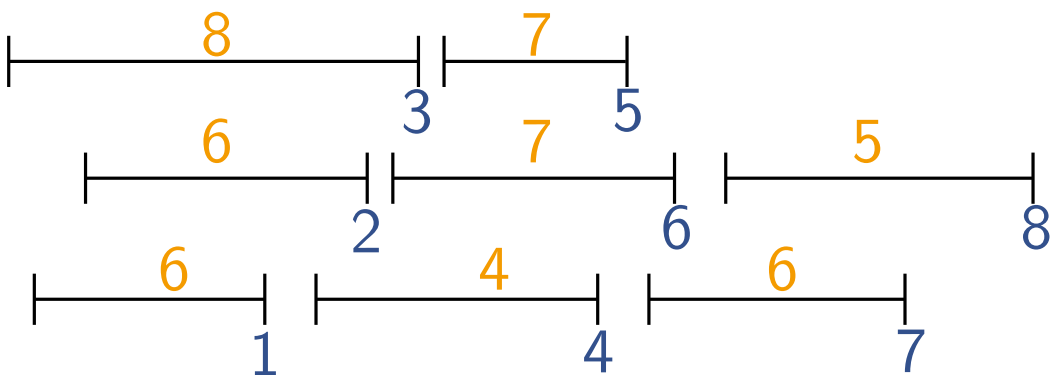
$T[i]$= Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + \max\{T[j] \mid j < i \text{ und } r(j) < l(i)\}\}$

$l(i)/r(i) =$ linker/rechter Endpunkt von Interval i

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]$= Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

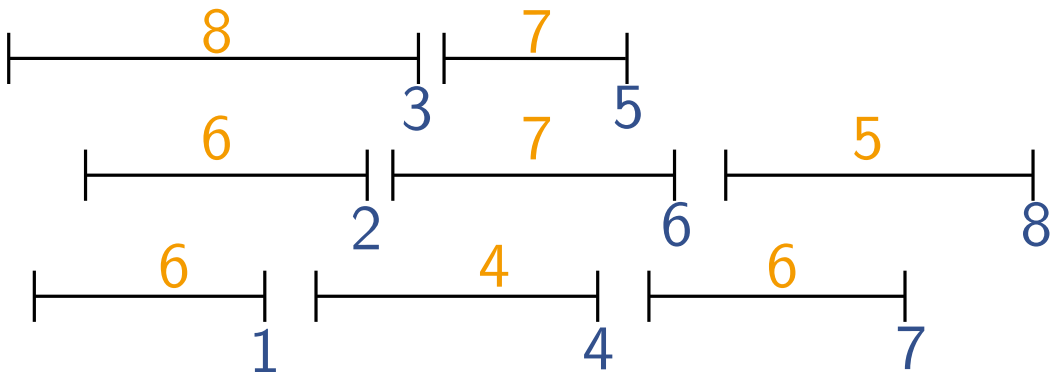
$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + \max\{T[j] \mid j < i \text{ und } r(j) < l(i)\}\}$

Laufzeit?

$l(i)/r(i) =$ linker/rechter Endpunkt von Interval i

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]$= Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

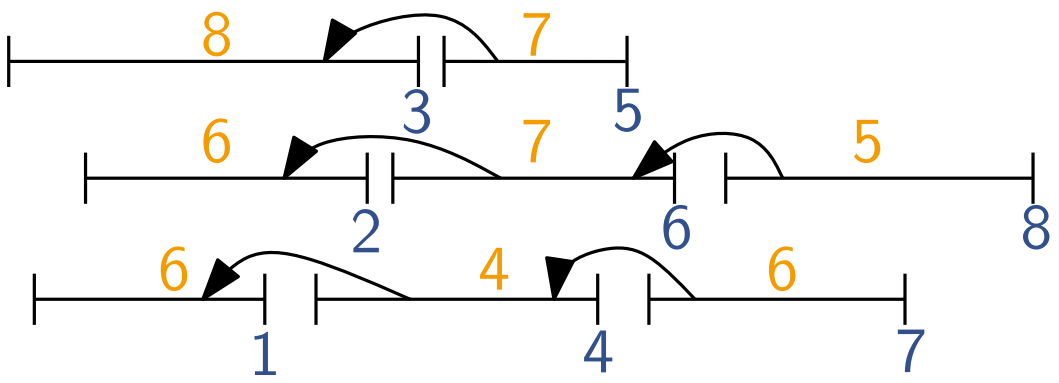
$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + \max\{T[j] \mid j < i \text{ und } r(j) < l(i)\}\}$

$O(n^2)$: Wie kann Laufzeit auf $O(n)$ verbessert werden?

$l(i)/r(i) =$ linker/rechter Endpunkt von Interval i

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

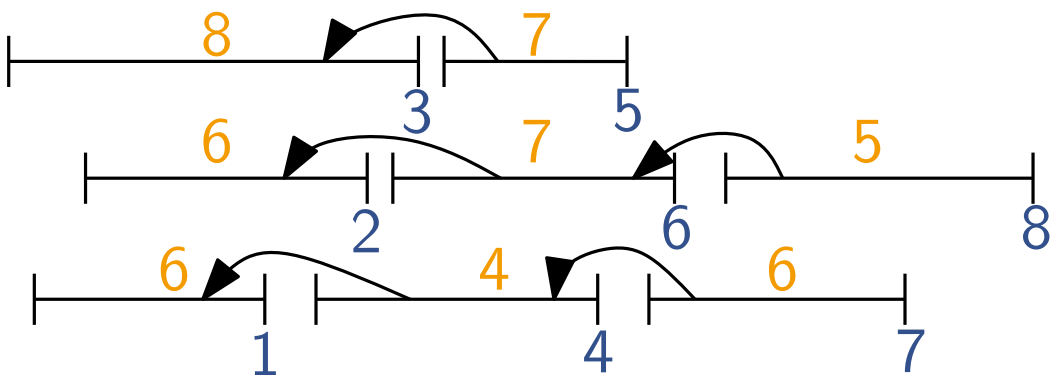
Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

Berechne für jedes Intervall i Vorgänger $pre(i)$ in $O(n)$ Zeit.

$pre(i) =$rechtestes Interval mit rechten Endpunkt links von Interval i.

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

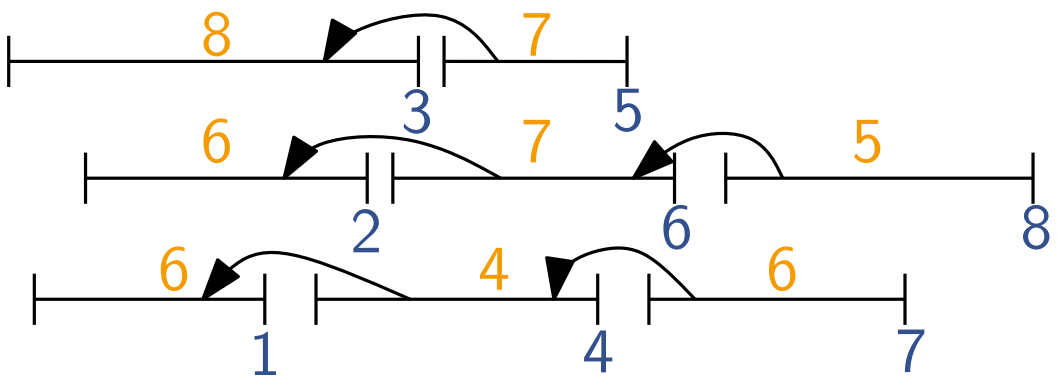
Berechne für jedes Intervall i Vorgänger $pre(i)$ in $O(n)$ Zeit.

$pre(i) =$rechtestes Interval mit rechten Endpunkt links von Interval i.

$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + T[pre(i)]\}$

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]=$ Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

Berechne für jedes Intervall i Vorgänger $pre(i)$ in $O(n)$ Zeit.

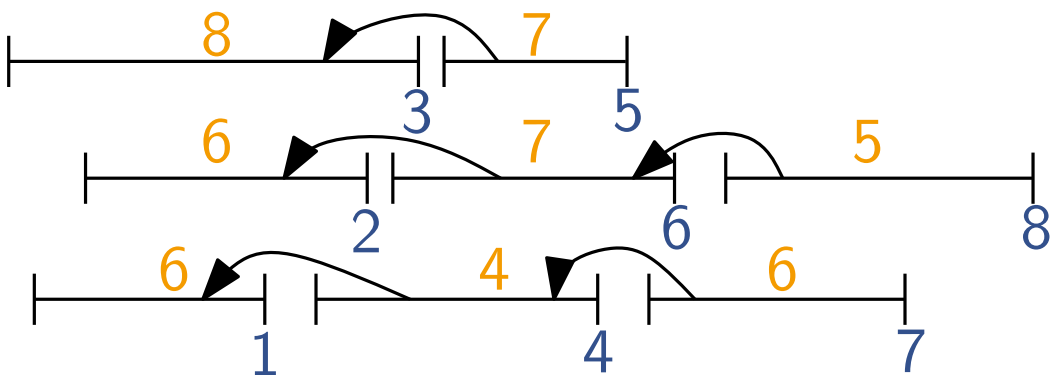
$pre(i) =$rechtestes Interval mit rechten Endpunkt links von Interval i.

$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + T[pre(i)]\}$

Laufzeit $O(n)$.

Lösung



Index des Intervals sortiert nach rechtem Endpunkt.

Gewicht des Intervals i: w(i)

Beobachtung: Gehört Interval i zur optimalen Lösung, dann trennt i Instanz in zwei unabhängige Teile auf.

$T[i]$= Wert der optimalen Lösung für Intervalle $1, \ldots, i$.

Berechne für jedes Intervall i Vorgänger $pre(i)$ in $O(n)$ Zeit.

$pre(i)$ =rechtestes Interval mit rechten Endpunkt links von Interval i.

$T[0] := 0$

$T[i] := \max\{T[i-1], w(i) + T[pre(i)]\}$

Laufzeit $O(n)$.

Annahme: Intervalle sind bereits sortiert.

k-RestrictedMaxTotal

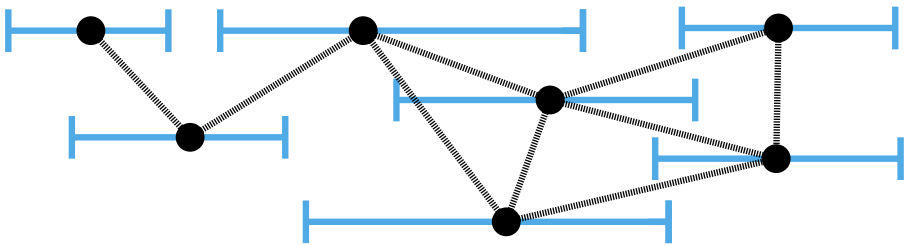
Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 1}$: Problem ist äquivalent zu *maximum weighted independent set* in einem **Intervallgraph**.

$\Rightarrow O(n)$ Zeit falls Intervalle sortiert sind

[Hsiao et al., 1992]



Gewicht eines Knoten = Länge eines Intervals

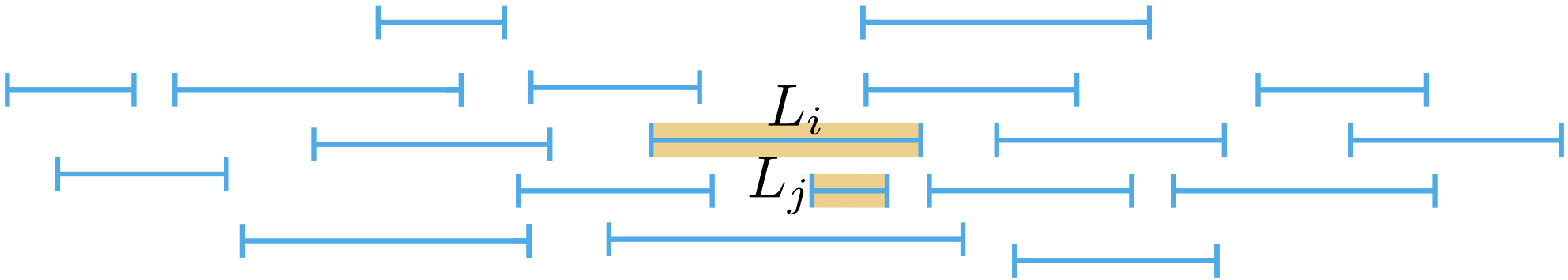
s. Übungsblatt

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$: Wegen **Konflikte** schwieriger als Fall $k = 1$

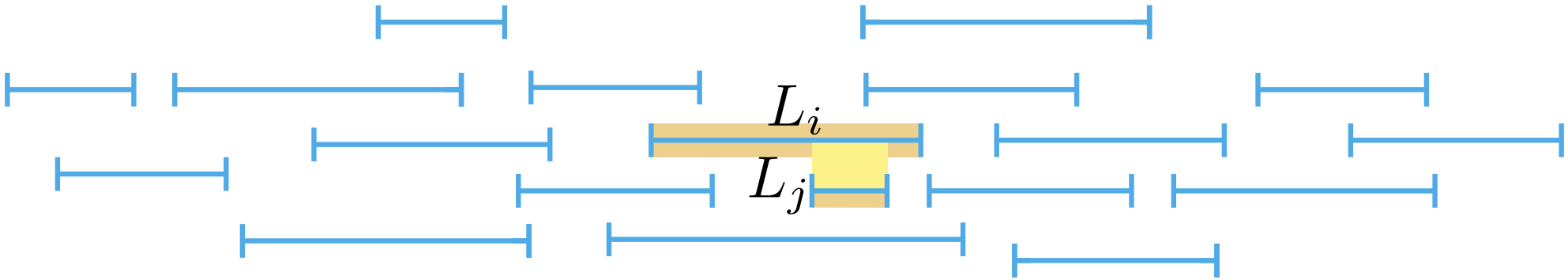


k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$: Wegen **Konflikte** schwieriger als Fall $k = 1$

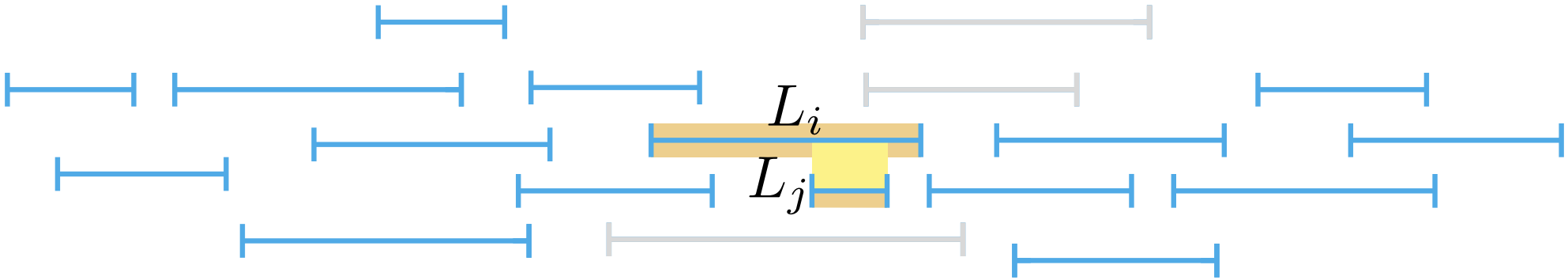


k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$: Wegen **Konflikte** schwieriger als Fall $k = 1$

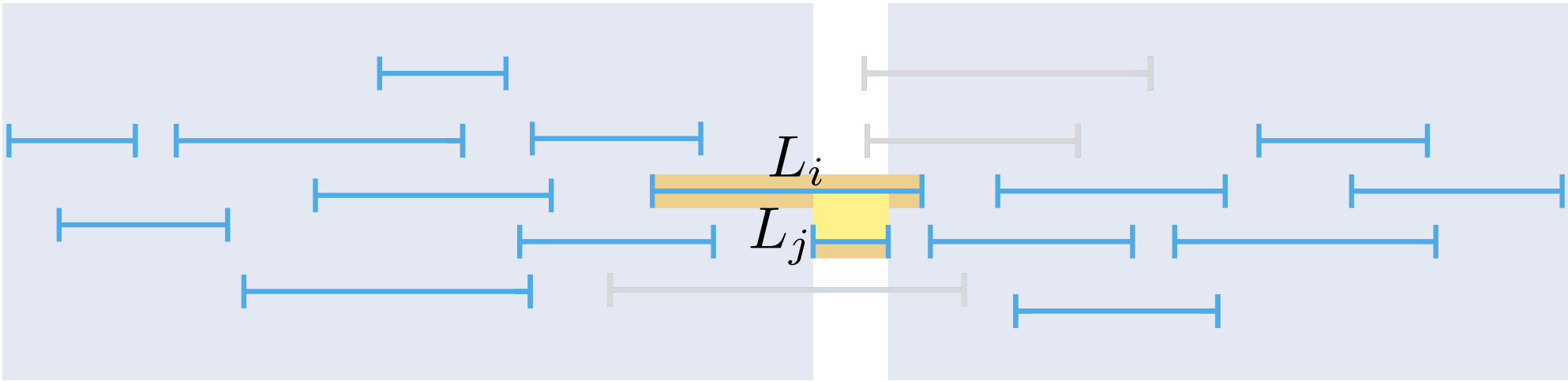


k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$: Wegen **Konflikte** schwieriger als Fall $k = 1$



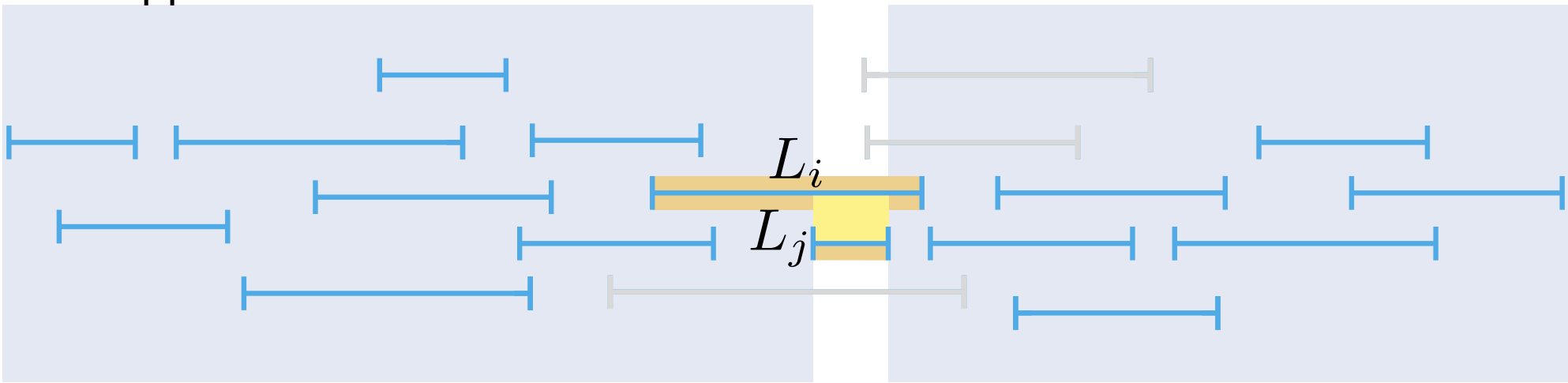
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$: Wegen **Konflikte** schwieriger als Fall $k = 1$

Ein **separierendes Tupel** sind zwei nicht im Konflikte stehende, sich überlappende Intervalle.



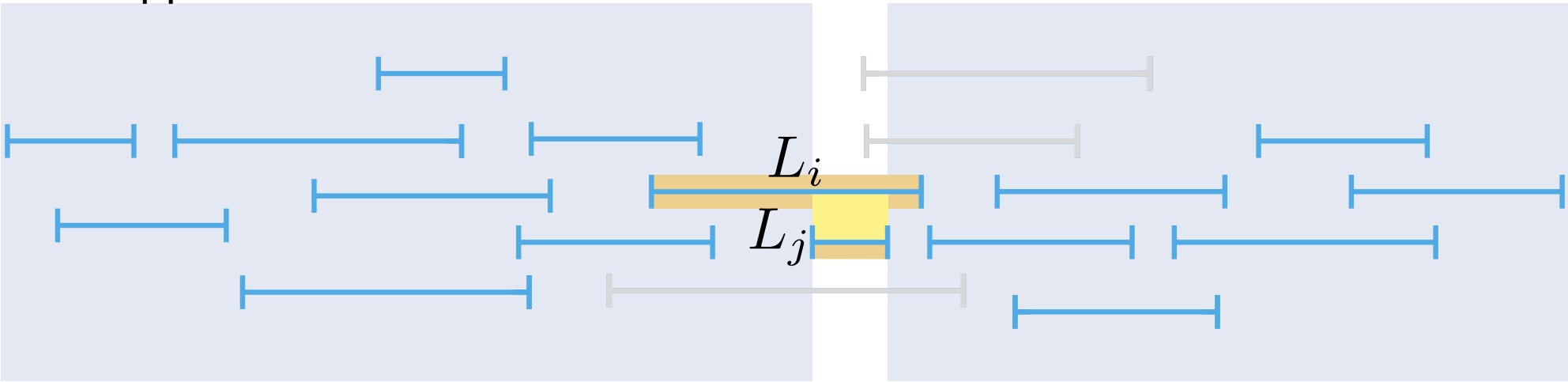
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$: Wegen **Konflikte** schwieriger als Fall $k = 1$

Ein **separierendes Tupel** sind zwei nicht im Konflikte stehende, sich überlappende Intervalle.



Beobachtung:
Ein separierendes Tupel teilt die Instanz in zwei Teilinstanzen.

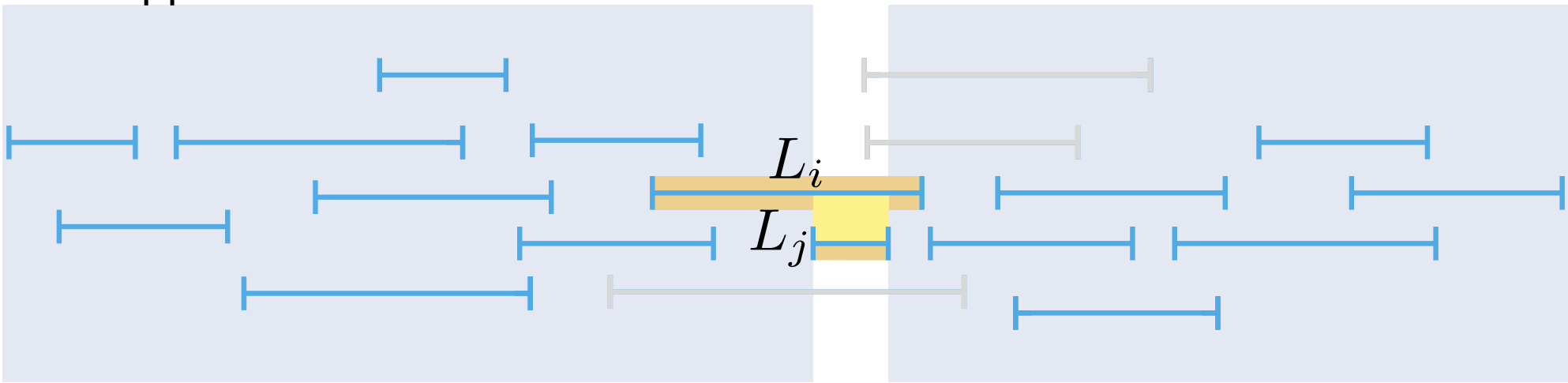
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$: Wegen **Konflikte** schwieriger als Fall $k = 1$

Ein **separierendes Tupel** sind zwei nicht im Konflikte stehende, sich überlappende Intervalle.



Beobachtung:
Ein separierendes Tupel teilt die Instanz in zwei Teilinstanzen.

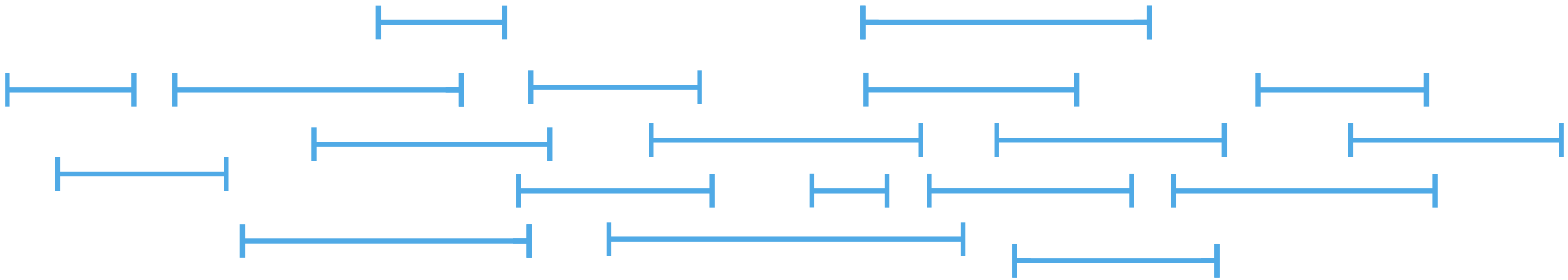
Dynamische Programmierung

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

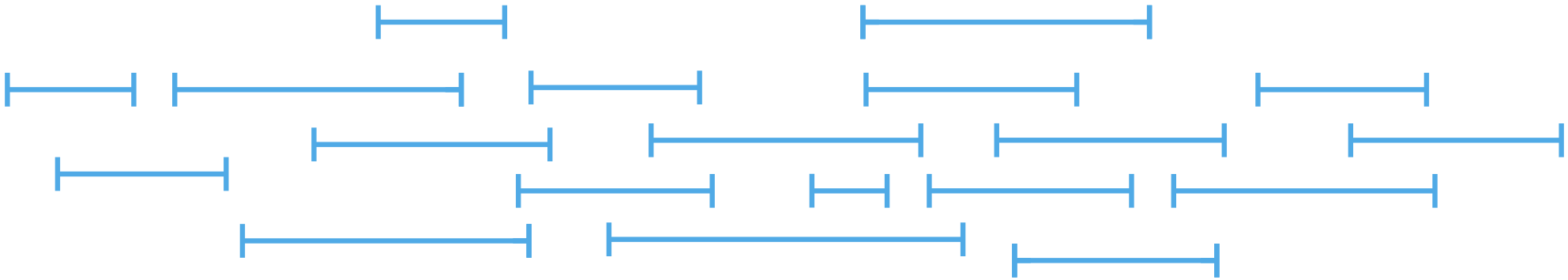


k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:



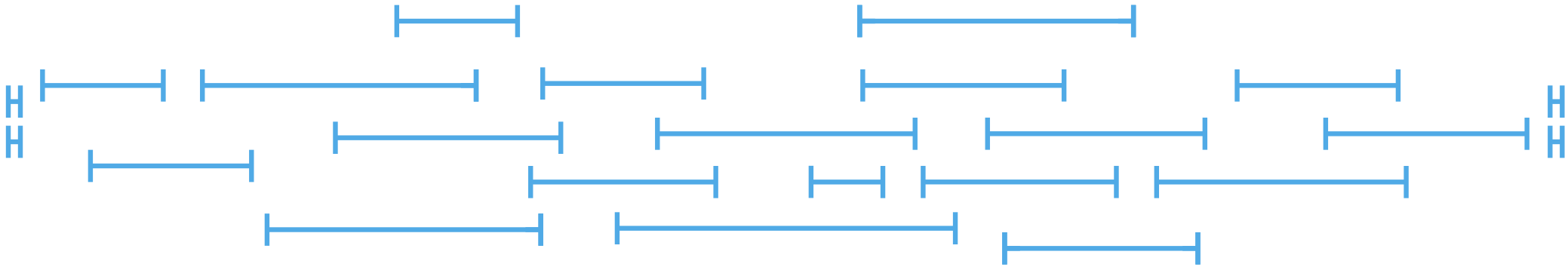
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.



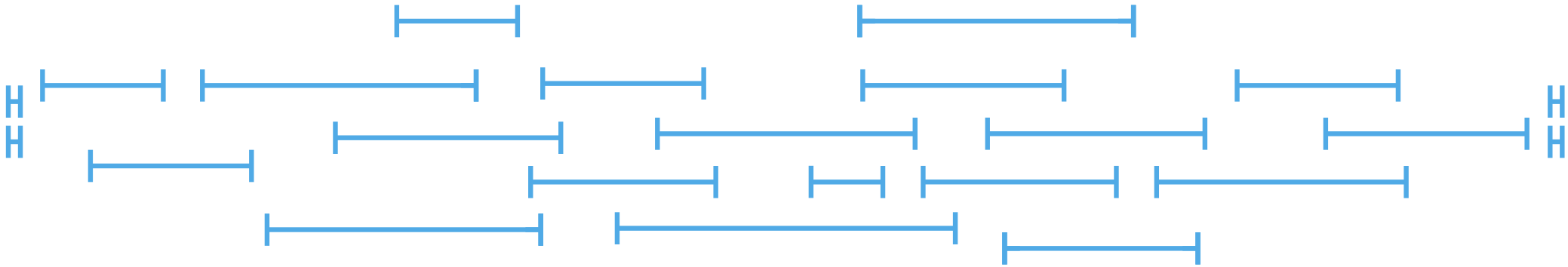
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



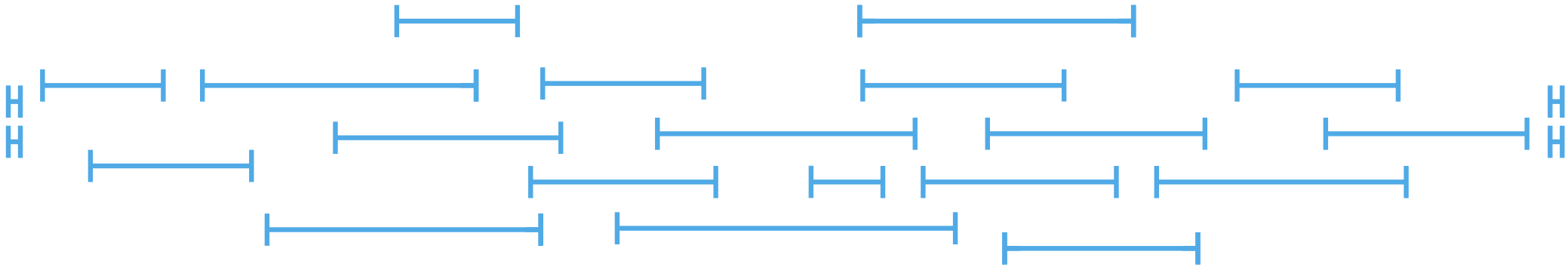
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

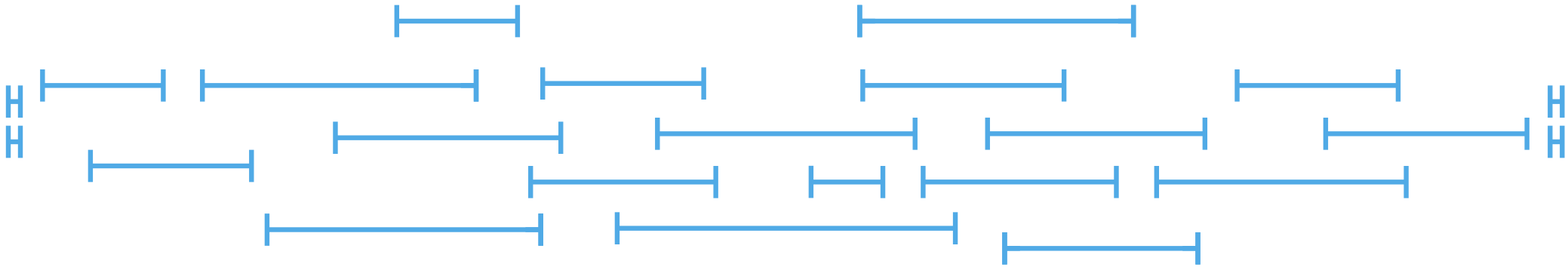
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

Konstruiere $\mathcal{T}$ von links nach rechts durch Raten des *vorherigen* separierenden Tupels.

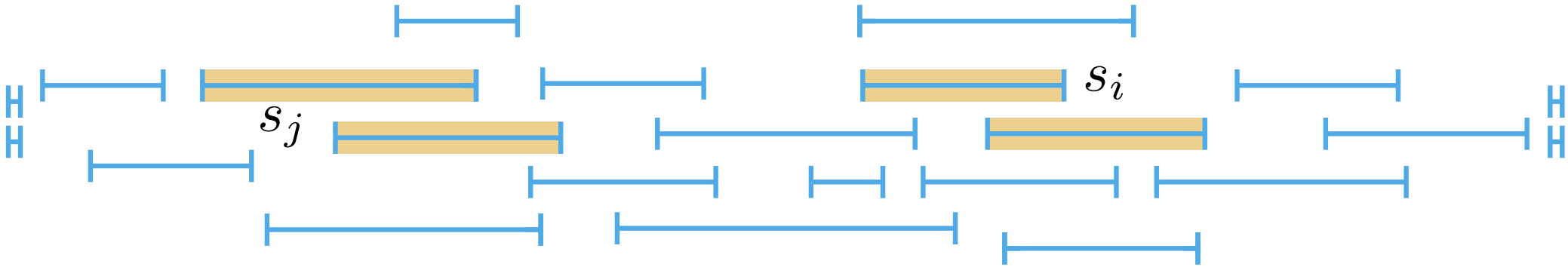
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

Konstruiere $\mathcal{T}$ von links nach rechts durch Raten des *vorherigen* separierenden Tupels.

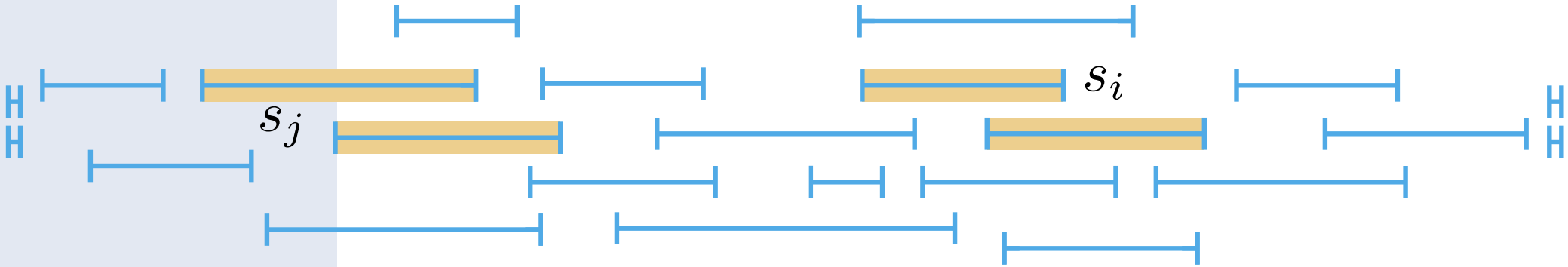
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

Konstruiere $\mathcal{T}$ von links nach rechts durch Raten des *vorherigen* separierenden Tupels.

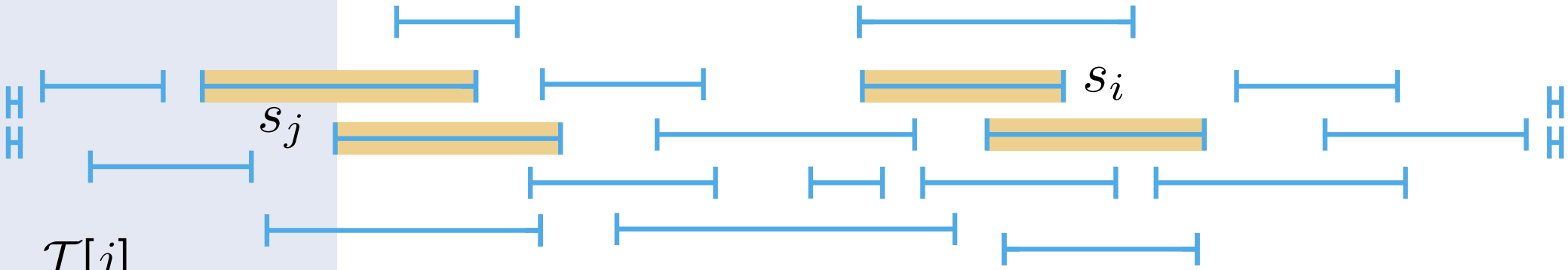
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

Konstruiere $\mathcal{T}$ von links nach rechts durch Raten des *vorherigen* separierenden Tupels.

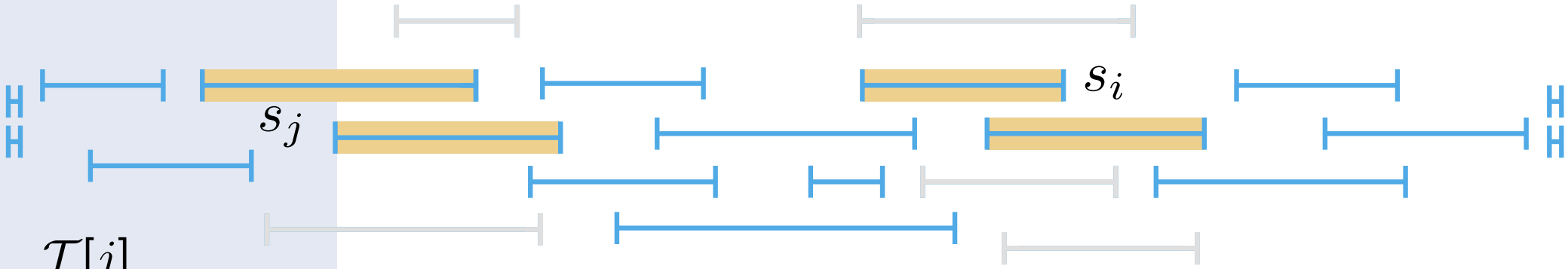
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

Konstruiere $\mathcal{T}$ von links nach rechts durch Raten des *vorherigen* separierenden Tupels.

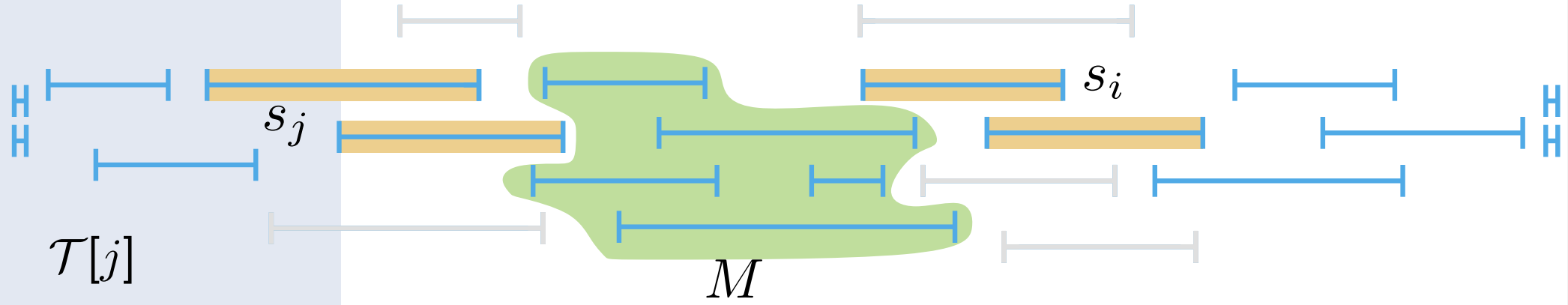
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

Konstruiere $\mathcal{T}$ von links nach rechts durch Raten des *vorherigen* separierenden Tupels.

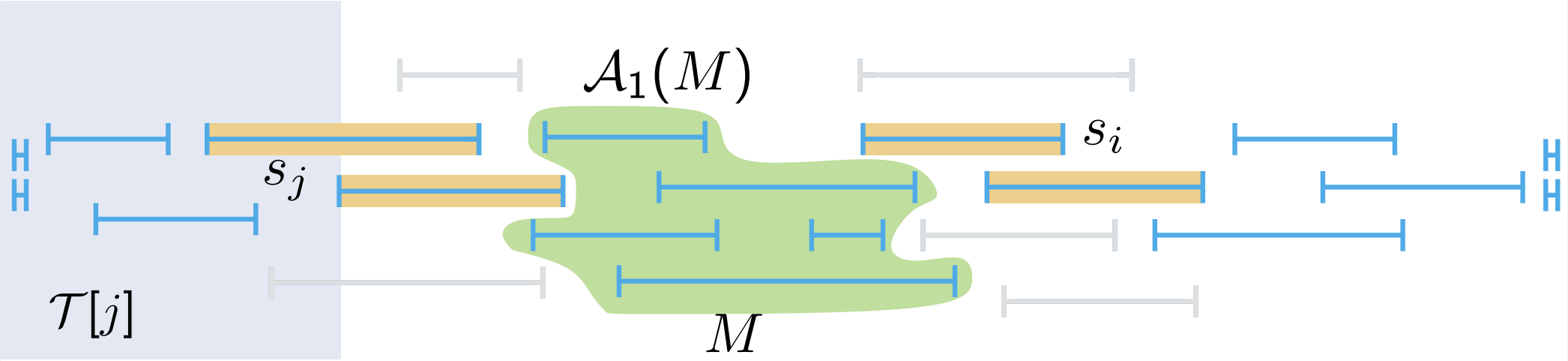
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

- Füge zwei separierende *Dummy*-Tupel hinzu.
- nummeriere alle separierende Tupel s_i und sortiere sie
- verwalte eindimensionale Tabelle $\mathcal{T}$



$\mathcal{T}[i]$: optimale Lösung für alle $s_j, j < i$, die s_i beinhalten.

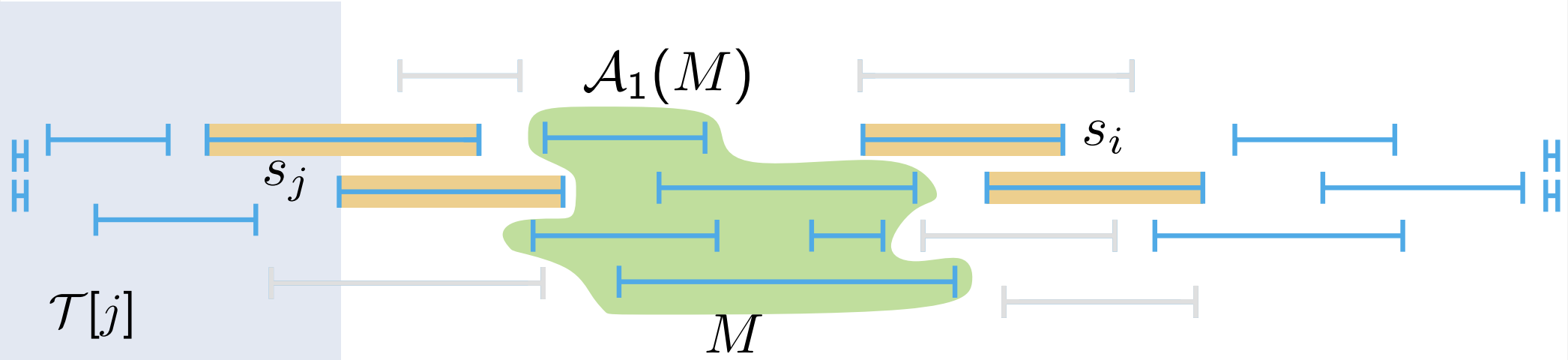
Konstruiere $\mathcal{T}$ von links nach rechts durch Raten des *vorherigen* separierenden Tupels.

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

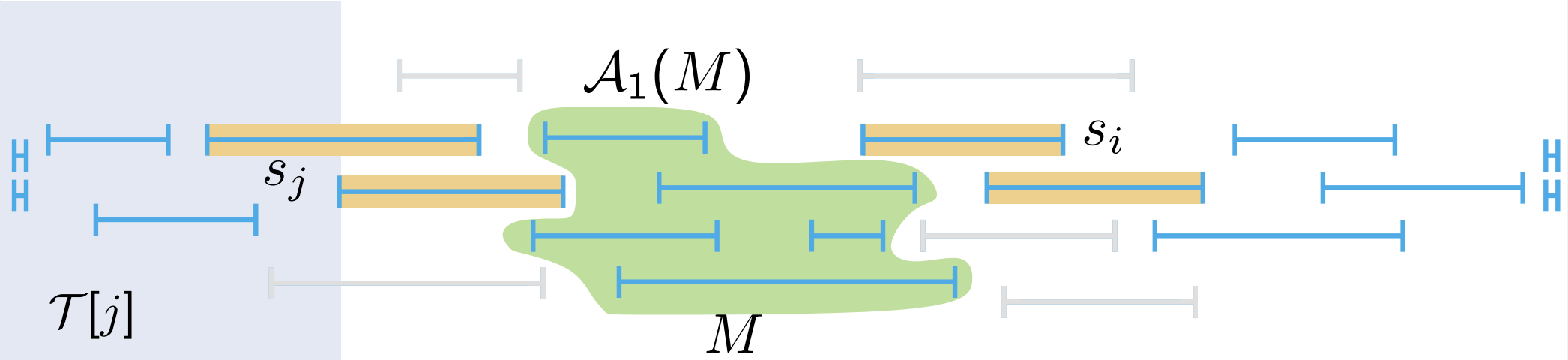


k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:



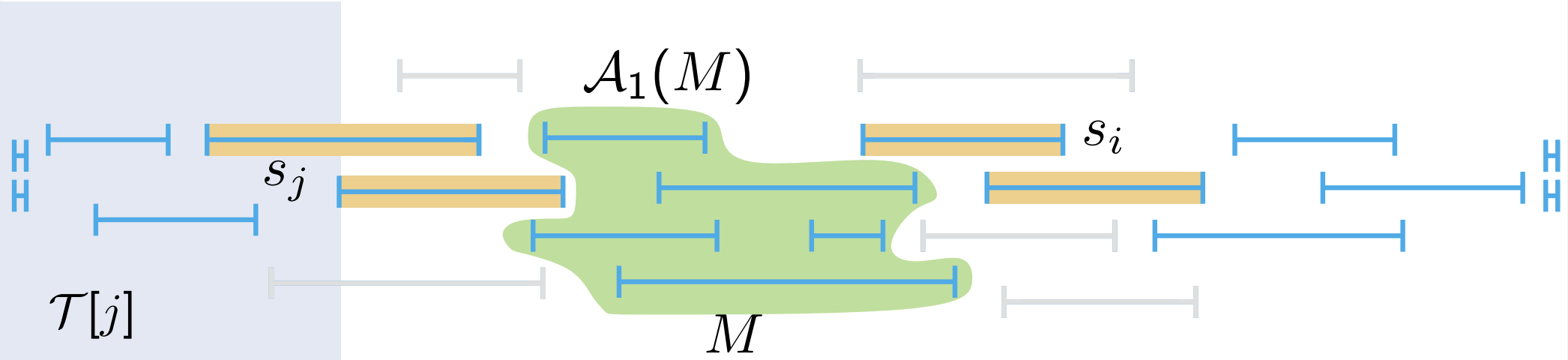
$$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_1(M) + \text{length}(s_i) \mid s_i, s_j \text{ kompatibel } \}$$

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:



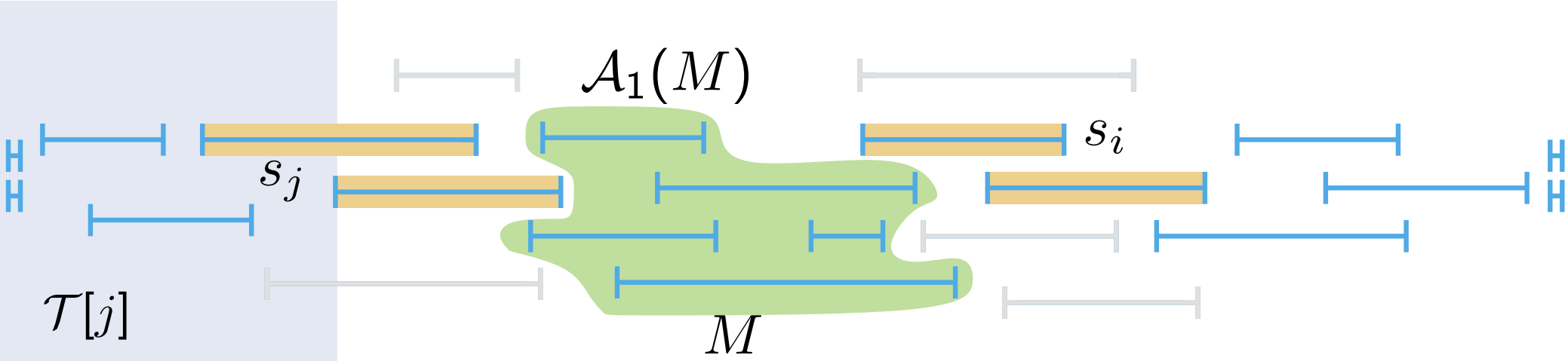
$$T[i] = \max_{j<i}\{\underbrace{T[j]}_{O(1)} + \underbrace{\mathcal{A}_1(M)}_{O(n)} + \underbrace{\text{length}(s_i)}_{O(1)} \mid s_i, s_j \text{ kompatibel } \}$$

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:



$$T[i] = \max_{j<i}\{\underbrace{T[j]}_{O(1)} + \underbrace{\mathcal{A}_1(M)}_{O(n)} + \underbrace{\text{length}(s_i)}_{O(1)} \mid s_i, s_j \text{ kompatibel } \}$$

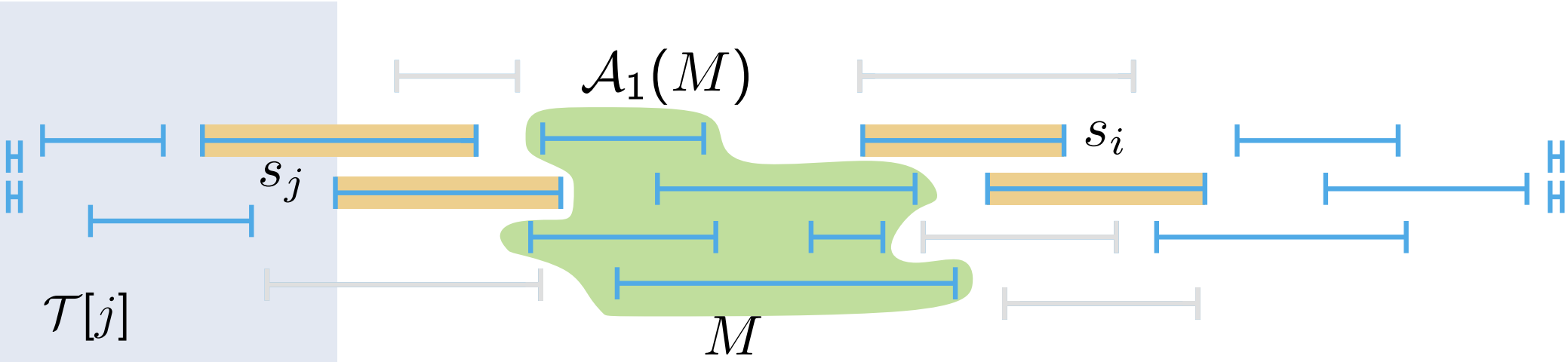
$O(\#\text{sep. tu.})$

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:



$$T[i] = \max_{j<i}\{\underbrace{T[j]}_{O(\#\text{sep. tu.})} + \underbrace{\mathcal{A}_1(M)}_{O(1)} + \underbrace{\text{length}(s_i)}_{O(n)} \mid s_i, s_j \text{ kompatibel } \}$$

$O(1)$ $O(\#\text{sep. tu.})$

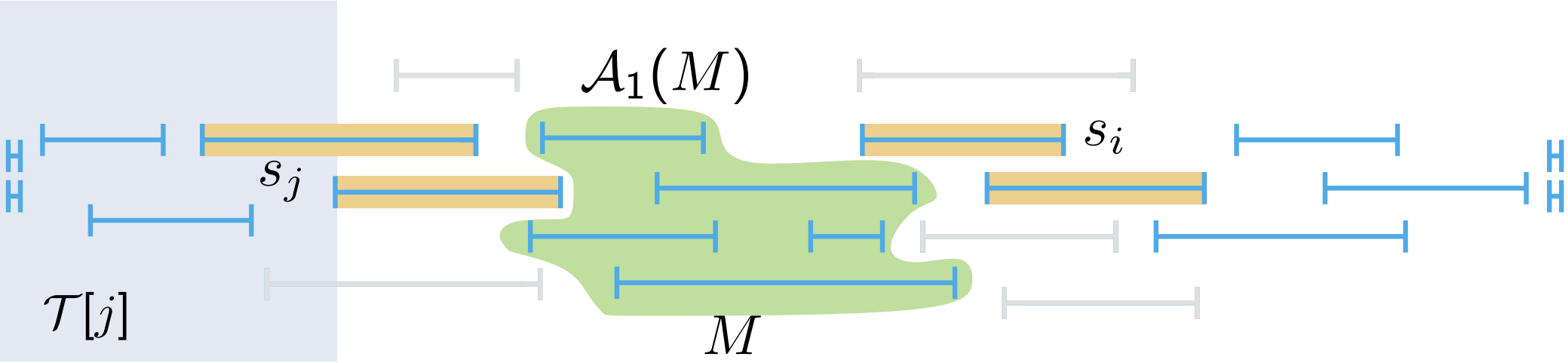
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

Zeitkomplexität: $O(n \cdot (\#\text{sep. tu.})^2)$



$O(\#\text{sep. tu.})$

$$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_1(M) + \text{length}(s_i) \mid s_i, s_j \text{ kompatibel } \}$$

$O(1)$ $O(n)$ $O(1)$

$O(\#\text{sep. tu.})$

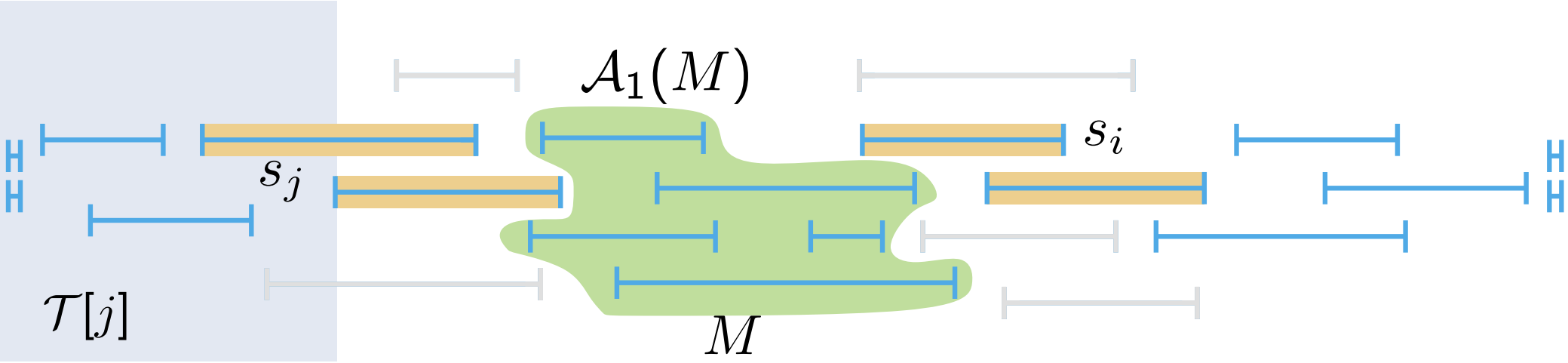
k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 2}$:

Zeitkomplexität: $O(n \cdot (\#\text{sep. tu.})^2) \quad = O(n^5)$



$O(\#\text{sep. tu.})$

$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_1(M) + \text{length}(s_i) \mid s_i, s_j \text{ kompatibel } \}$

$O(1) \quad O(n) \quad O(1)$

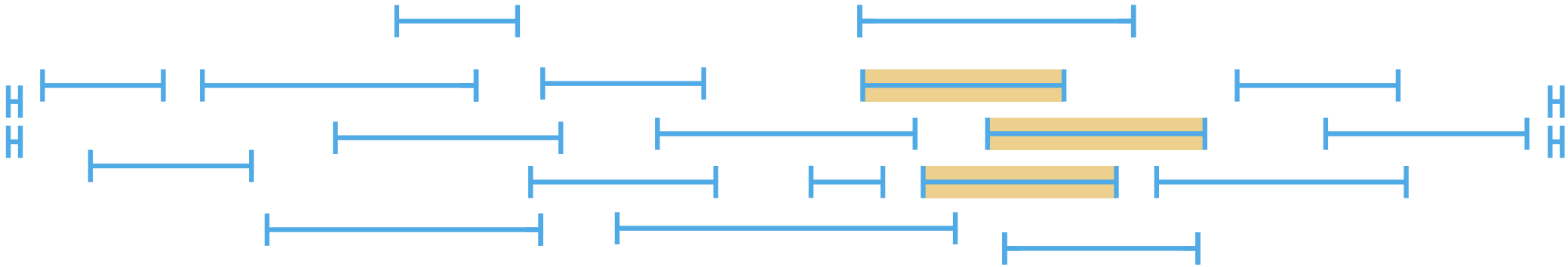
$O(\#\text{sep. tu.})$

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 3}$:

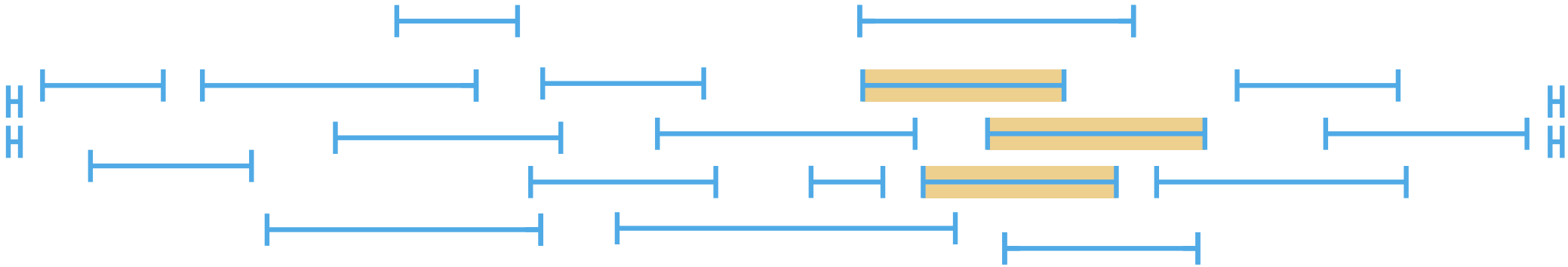


k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 3}$:



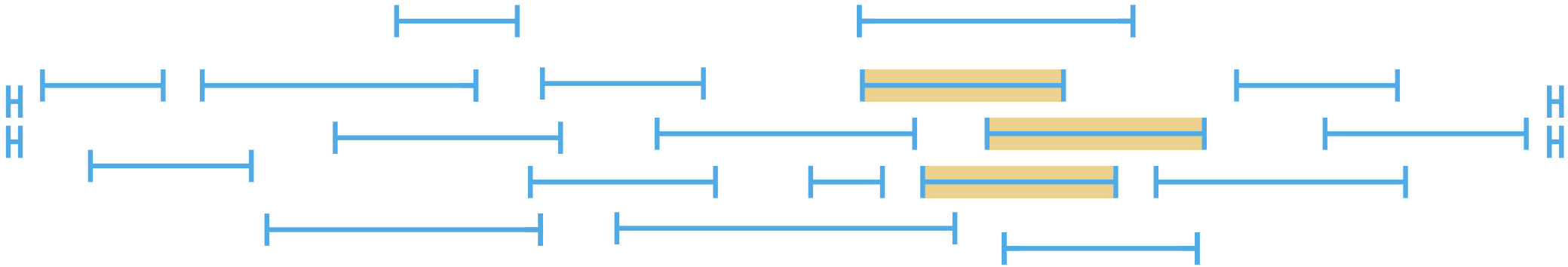
$$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_2(M) + \text{length}(s_i) \mid s_i, s_j \text{ kompatibel } \}$$

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k = 3}$:



Kann verallgemeinert werden zu:

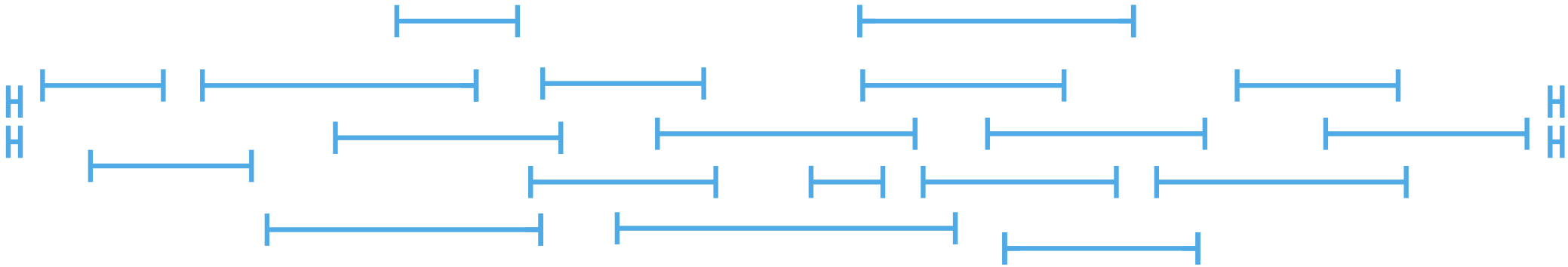
$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_2(M) + \text{length}(s_i) \mid s_i, s_j \text{ kompatibel } \}$

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k > 1}$:



Kann verallgemeinert werden zu:

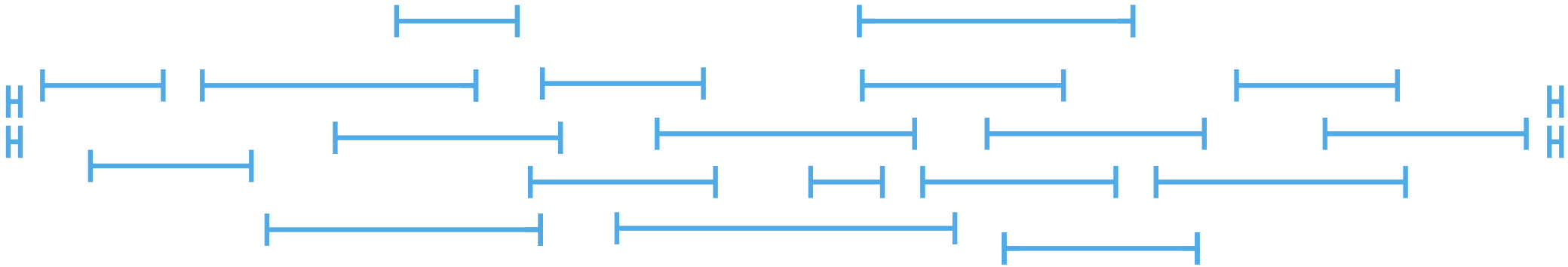
$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_{k-1}(M) + \text{length}(s_i) \mid s_i, s_j \text{ kompatibel } \}$

k-RestrictedMaxTotal

Theorem

k-RestrictedMaxTotal kann in polynomieller Zeit gelöst werden.

Fall $\mathbf{k > 1}$:



Kann verallgemeinert werden zu:

$T[i] = \max_{j<i}\{T[j] + \mathcal{A}_{k-1}(M) + \text{length}(s_i) \mid s_i, s_j \text{ kompatibel } \}$

Zeitkomplexität: $O(n^{k^2+k-1})$

Zusammenfassung

Ergebnisse:

- GeneralMaxTotal ist $\mathcal{NP}$-vollständig
- k-RestrictedMaxTotal is polynomiell lösbar
- neues, flexibles Beschriftungsmodell
- Algorithmen hängen nicht von der Geometrie ab.

Offene Fragen:

- k-RestrictedMaxTotal ist polynomiell lösbar, aber sehr langsam (wie schlecht für realistische Daten?)
- Trajectory-based Labeling mit **Zoomen**?