# LU Einführung in Wissensbasierte Systeme
# Aufgabenblatt 2: Planen

Michael Fink und Hans Tompits
Arbeitsbereich Wissensbasierte Systeme 184/3,
Technische Universität Wien

Wintersemester 2010/2011

15. November 2010

## Allgemeine Problemstellung

Die zweite Übungsaufgabe beschäftigt sich mit Planen. Als Planungsdomäne für diesen Übungsblock soll uns, ähnlich zu der in der Vorbesprechung zur Übung vorgestellten Blocks World, die Domäne des sogenannten *Bridge and Torch Problems* (auch *Bridge Crossing Problem (BCP)* genannt) dienen. Bei diesem Rätselproblem stehen vier Personen vor der Aufgabe einen Fluß bei Nacht zu überqueren. Dazu steht eine Brücke zur Verfügung, welche maximal zwei Personen gleichzeitig überqueren können. Dazu muß außerdem eine Fackel benutzt werden, von der es in der klassischen Variante des Rätsels genau eine gibt, und diese befindet sich zu Beginn am Ausgangsufer. Die vier Personen sind unterschiedlich schnell beim Überqueren der Brücke. Wenn zwei Personen gleichzeitig die Brücke überqueren, so ist das Tempo ihres Vorwärtskommens durch die langsamere der beiden Personen bestimmt. Die Frage lautet, ob es den vier Personen möglich ist in einer vorgegeben Zeit (oder schneller) ans gegenüber gelegene Ufer zu gelangen.

Ziel dieses Übungsblocks ist die Formalisierung verschiedener Varianten dieses Rätsels als Planungsdomäne und das Lösen der Rätsel durch das Berechnen von Plänen. Halten Sie sich bei Ihren Lösungen immer an die in diesem Aufgabenblatt vorgegebenen Namenskonventionen.

## Aufgabe 2.1: Das Bridge and Torch Problem

Betrachten wir zunächst eine klassische Variante des BCP. Die Personen seien ein Ritter (`knight`), eine Dame (`lady`), ein König (`king`) und eine Königin (`queen`). Ihre Geschwindigkeit sei in Minuten wie folgt gegeben: Der Ritter benötigt $1$ Minute, die Dame $2$ Minuten, der König $5$ Minuten und die Königin $8$ Minuten. Implementieren Sie dieses Rätsel als Planungsproblem mit dem System $\mathtt{DLV}^{\mathcal{K}}$, wie in den Folien zur Einführungsveranstaltung der Laborübung anhand der Blocks World Domäne exemplifiziert. Gehen Sie dabei wie folgt vor.

### Hintergrundwissen

Modellieren Sie das Hintergrundwissen der oben beschriebenen Domäne in einem Answer-Set Programm und befolgen Sie dabei nachstehende Namenskonventionen:

(h1) Benutzen Sie ein unäres Prädikat `location` und die Konstanten `here` und `there` um die beiden Ufer des Flusses zu repräsentieren.

(h2) Verwenden Sie ein unäres Prädikat `person` und die Konstanten `knight`, `lady`, `king` und `queen` um die handelnden Personen, sowie ein binäres Prädikat `speed`$(\mathtt{P}, \mathtt{S})$ um deren Geschwindigkeiten wie gegeben zu repräsentieren, wobei `P` eine Person ist und `S` für einen ganzahligen Wert (die jeweils benötigten Minuten) steht.

(h3) Außerdem ist bekannt, das der Ritter schon etwas betagt ist. Repräsentieren Sie diese Information mit Hilfe eines unären Prädikates `aged`. (Diese Information wird erst in später folgenden Varianten des Rätsels benutzt, muß aber von Beginn an, d.h. auch für diese erste Aufgabe, im Hintergrundwissen repräsentiert werden.)

(h4) Wir werden zur Lösung des Problems mit unterschiedlichen Zeitwerten (Dauer in Minuten) hantieren müssen. Benutzen Sie das unäre Prädikat `duration`, um den entsprechenden Wertebereich gültiger Werte (für entsprechende Typdeklarationen) zu definieren. Es gilt, dass jeder ganzzahlige Wert von $0$ bis $60$ (jeweils einschließlich) eine mögliche Zeitdauer ist.

Speichern Sie das Hintergrundwissen in einer Datei namens `bcbackground.dl` ab. Rufen Sie `DLV` mittels `dlv -N=60 bcbackground.dl` auf, um die Answer Sets des Hintergrundwissens zu berechnen. Vergewissern Sie sich, dass Sie genau ein Answer Set erhalten, welches das Hintergrundwissen korrekt repräsentiert.

## Planungsdomäne

Modellieren Sie nun die Planungsdomäne in $\mathcal{K}$ und gehen Sie dabei wie folgt vor:

(d1) Deklarieren Sie ein binäres Fluent `at`$(\mathtt{P}, \mathtt{L})$, dessen erstes Argument eine Person und dessen zweites Argument ein Ort, d.h. ein Ufer, ist, sowie ein unäres Fluent `torch`$(\mathtt{L})$ mit einem Ort als Argument. Die beiden Fluents sollen im Weiteren dazu dienen, den jeweiligen Aufenthaltsort der Personen bzw. der Fackel an einem der beiden Ufer zu beschreiben.

Deklarieren Sie darüber hinaus zwei unäre Fluents `elapsed`$(\mathtt{T})$ und `timeout`$(\mathtt{T})$, welche beide eine Zeitdauer als Argument haben und die verstrichene Zeit bzw. eine Zeitschranke repräsentieren.

(d2) Deklarieren Sie eine Aktion `cross_single`$(\mathtt{P}, \mathtt{T}, \mathtt{L})$, welche das Überqueren der Brücke einer einzelnen Person darstellt. Dabei erfaßt das erste Argument die Person, das zweite Argument die Dauer des Überquerens und das dritte Argument repräsentiert ein Ufer (jenes zu dem übergesetzt wird; im Folgenden als das "Zielufer" der Aktion bezeichnet).

Deklarieren Sie weiters eine Aktion `cross_together`$(\mathtt{P1}, \mathtt{P2}, \mathtt{T}, \mathtt{L})$, welche das gemeinsame Überqueren der Brücke zweier Personen repräsentiert, wobei `P1` und `P2` die beiden Personen sind. Bei den Argumenten `T` und `L` handelt es sich wiederum um eine Zeitdauer und einen Ort (Zielufer).

(d3) Spezifizieren Sie entsprechende Ausführbarkeitsbedingungen für die beiden Aktionen `cross_single` und `cross_together`.

Die Voraussetzungen dafür sind wie folgt. In beiden Fällen müssen sich ($a$) die jeweilige(n) Person(en) und die Fackel an jenem Ufer befinden, welches dem Zielufer gegenüber liegt (also jenem von dem aus übergesetzt wird; im Weiteren als "Startufer" der Aktion bezeichnet).

Darüber hinaus ist ($b$) eine Aktion `cross_single` nur dann ausführbar, wenn die entsprechende Zeitdauer gleich der Geschwindigkeit jener Person ist, welche die Brücke

überquert. Für die Dauer einer Aktion `cross_together` gilt die Bedingung ($c$), dass diese gleich der Geschwindigkeit der langsameren der beiden querenden Personen ist.

Um sicher zu stellen, dass zwei unterschiedliche Personen an einer `cross_together` Aktion beteiligt sind, sowie um symmetrische `cross_together` Aktionen zu verhindern, ist dafür noch folgende Ausführbarkeitsbedingung ($d$) zu realisieren:
`cross_together` ist nur dann ausführbar, wenn `P1` $>$ `P2` gilt.

(Hier wird die Tatsache benutzt, dass auch nichtnumerische Konstanten in DLV$^{\mathcal{K}}$ miteinander vergleichbar sind. Nehmen Sie beispielsweise an, dass `queen` $>$ `king` gilt, dann ist unter Bedingung ($d$) von den Aktionen `cross_together`(`king`, `queen`, 8, `there`), `cross_together`(`queen`, `queen`, 8, `there`) und `cross_together`(`queen`, `king`, 8, `there`) nur die letztere ausführbar.)

(d4) Implementieren Sie die Effekte der Aktionen `cross_single` und `cross_together`.

Außer ($a$) dem Aufenthaltsort der Fackel, der sich durch das Ausführen der jeweiligen Aktion verändert, ändert sich dadurch auch ($b$) der Aufenthaltsort der beteiligten Personen und ($c$) die verstrichene Zeit, wenn die Zeitdauer der Aktion größer $0$ ist (erhöht sich um die Zeitdauer der Aktion).

**Achtung:** Vergessen Sie nicht, dass ein neuer Aufenthaltsort bzw. ein neuer Wert für die verstrichene Zeit, immer auch bedeutet, dass der bisherige Aufenthaltsort bzw. der bisherige Zeitwert nicht mehr gilt.

(d5) Der Aufenthaltsort einer Person sowie der Fackel an einem der beiden Ufer soll sich nicht grundlos, sondern *nur* durch das Ausführen von Aktionen (Brückenüberquerungen) mit entsprechenden Effekten verändern. Gleiches gilt für die verstrichene Zeit und die Zeitschranke. D.h., die entsprechenden Fluents `at`, `torch`, `elapsed` und `timeout` unterliegen der Inertia. Modellieren Sie das entsprechend.

(d6) Zuletzt sind statische Einschränkungen bezüglich gültiger Zustände entsprechend zu implementieren. Insbesondere, dass weder eine Person noch die Fackel gleichzeitig an zwei Orten sein kann. Außerdem muß der Wert der verstrichenen Zeitdauer (`elapsed`) immer eindeutig sein und er darf nicht größer als die Zeitschranke (`timeout`) sein.

Vergessen Sie nicht dafür zu sorgen, dass Aktionen nur sequentiell ausführbar sind, und speichern Sie die Planungsdomäne in einer Datei namens `bcdomain.plan` ab.

## Planungsproblem

Nun ist nur noch das konkrete Planungsproblem, die konkrete Aufgabenstellung des Rätsels, zu repräsentieren. Speichern Sie den entsprechenden Anfangszustand (alle vier Personen und die Fackel am Ausgangsufer (`here`); noch keine Zeit verstrichen und maximal $15$ Minuten Zeit) und die Zielbedingung (alle vier Personen am gegenüber liegenden Ufer (`there`)) in einer Datei namens `bcproblem1.plan`.

Starten Sie DLV$^{\mathcal{K}}$ mittels

```
dlv -FPopt bcbackground.dl bcdomain.plan bcproblem1.plan
    -N=60 -planlength=n
```

wobei `n` die maximale Planlänge (Anzahl der Aktionen), die der berechnete Plan haben soll, darstellt. DLV$^{\mathcal{K}}$ gibt alle gefundenen Pläne mit der Länge `n` aus (falls es kürzere Pläne gibt, die mittels `no action` zu Plänen der Länge `n` werden, so sind diese natürlich enthalten). Insbesondere gibt DLV$^{\mathcal{K}}$ die Beschreibung der einzelnen Zustände (Zeilen mit

STATE: beginnend), die Aktionen, sowie den Gesamtplan (Zeilen mit PLAN: beginnend) aus. Wenn Sie die Ausgabe auf Pläne reduzieren wollen, dann können Sie auf POSIX kompatiblen Systemen, beispielsweise unter Linux im Informatiklabor, dafür das Kommando grep verwenden (einfach | grep -c PLAN an obigen Aufruf anhängen).

**Achtung:** Verwenden Sie (wie oben) den Kommandozeilenparameter -N, um einen Maximalwert für die Integerarithmetik zu setzen (und keinesfalls #maxint).

**Hinweis:** Testen Sie Ihre Modellierung auch mit anderen Start- und Zielvorgaben auf Korrektheit!

## Aufgabe 2.2: Bridge and Torch Variante

Wenden wir uns nun einer Variante des BCP zu, bei der die Fackel nicht unbedingt am Ausgangsufer zur Verfügung steht. Wenn dies nicht der Fall ist, dann ist mit den bisherigen Aktionen auch kein Überqueren der Brücke möglich. Deshalb gibt es zusätzlich die Möglichkeit, dass eine einzelne Person die Brücke im Dunkeln überquert. Dies ist allerdings nur mutigen Personen möglich und sie benötigen dazu dreimal so lange als mit der Fackel. Personen von denen bekannt ist, dass sie gesund sind und die nicht betagt sind (oder zumindest nicht bekannt ist, dass sie betagt sind) sind mutig. Unabhängig davon ist der König mutig wannimmer die Königin anwesend (mit ihm am selben Ufer) ist. Darüber hinaus hat die Dame die Fähigkeit Personen die krank sind zu heilen (außer sich selbst), sofern sie eine Zeitspanne mit der kranken Person am gleichen Ufer verweilt.

Ein entsprechende Problemstellung ergibt sich beispielsweise aus der vorherigen Problemstellung, modifiziert und erweitert um die Tatsachen, dass

1. sich die Fackel zu Beginn am gegenüberliegenden Ufer befindet,
2. bekannt ist dass der Ritter, die Dame und die Königin zu Beginn gesund sind, während über die Gesundheit des Königs nichts bekannt ist,
3. nunmehr 23 Minuten zum Überqueren zur Verfügung stehen.

Implementieren Sie auch dieses Rätsel als Planungsproblem und erweitern Sie dazu Ihre bisherige Lösung zu Aufgabe 2.1 wie folgt.

- **Planungsdomäne**: Erweitern Sie die Modellierung der Planungsdomäne um die folgenden Aspekte und speichern Sie diese in der Datei bctorch.plan ab. (Nur neu hinzugekommene causal laws in der Datei speichern und diese zusammen mit der Datei bcdomain.plan verwenden; siehe auch dlv Aufruf unten!)

  (o1) Deklarieren Sie zwei unäre Fluents healthy und brave, deren Argument eine Person ist. Diese beiden Fluents sollen die Gesundheit einer Person, bzw. die Tatsache ob die jeweilige Person gerade mutig ist, modellieren.

  (o2) Deklarieren und implementieren Sie die Aktion cross_dark(P, T, L), welche das Überqueren der Brücke einer einzelnen Person (P) im Dunkeln repräsentiert, wobei das zweite und dritte Argument wiederum eine entsprechende Zeitdauer und das Zielufer der Aktion darstellen.

  Dabei soll folgendes gelten:

  * Die Aktion ist ausführbar, wenn sich die Person am Startufer der Aktion und die Fackel am Zielufer der Aktion befinden, wenn die Person mutig ist und wenn die Dauer gleich der dreifachen Dauer bei normaler Geschwindigkeit für die Person ist.
  * Nach dem Überqueren befindet sich die ausführende Person am Zielufer und es ist eine der Dauer der Aktion entsprechende Zeitspanne mehr verstrichen.

(o3) Modellieren Sie die statischen Abhängigkeiten (indirekten Effekte bez.) des Fluents `brave`, d.h., dass eine Person mutig ist, wenn sie gesund (`healthy`) ist und nicht bekannt ist ob bzw. dass sie betagt (`aged`) ist, und dass der König mutig ist, wannimmer er sich am gleichen Ufer wie die Königin befindet.

(o4) Modellieren Sie die Fähigkeit der Dame zu heilen, als folgenden einfachen dynamischen Zusammenhang: Eine andere Person als die Dame ist gesund wenn, nachdem sie sich gemeinsam mit der Dame an einem Ort befunden hat und bekannt war dass sie nicht gesund ist ($-$`healthy`), die Person sich im folgenden Zustand immer noch gemeinsam mit der Dame am selben Ort befindet. (Folglich kann die Dame bei einem derartigen Zustandsübergang auch mehr als eine Person heilen.)

Darüber hinaus, unterliegt das Wissen um die Gesundheit bzw. Krankheit einer Person (sowohl Fluent `healthy`, als auch Fluent $-$`healthy`) der Inertia.

- **Planungsproblem**: Modifizieren und erweitern Sie den Anfangszustand in der Datei `bcproblem1.plan` entsprechend den geänderten Bedingungen (Fackel `there`; Ritter, Dame und Königin gesund; 23 Minuten Zeit) und speichern Sie das modifizierte Problem in einer Datei namens `bcproblem2.plan`.

Verwenden Sie $\texttt{DLV}^{\mathcal{K}}$ mittels

```
dlv -FPopt bcbackground.dl bcdomain.plan bctorch.plan
    bcproblem2.plan -N=60 -planlength=n
```

Erneut gilt: Vergessen Sie nicht auf das Testen mit anderen Start- und Zielvorgaben!

## Aufgabe 2.3: BCP mit parallelen Aktionen

Wir wollen nun das BCP um eine weitere Variante bereichern: Es gibt ein Seil, welches mit einem Ende in der Mitte der Brücke befestigt ist und dessen anderes Ende an einem der beiden Ufer festgebunden ist. Eine gesunde Person kann das Seil benutzen, um sich an das gegenüberliegende Ufer zu schwingen. Dies kann aber immer nur gleichzeitig mit einer Überquerung der Brücke und in Richtung der Überquerung erfolgen. Die Dauer eines derartigen parallelen Schwunges ist vernachlässigbar, aber der Schwung beeinträchtigt die Gesundheit der ausführenden Person, wenn sich die Fackel nicht während des gesamten Schwungs (also auch bereits zu Beginn der Ausführung) am Zielufer befindet.

Ausgehend von der letzten Problemstellung (aus Aufgabe 2.2) und der Annahme, dass sich das Seil zu Beginn am Ausgangsufer befindet, kommt man beispielsweise zur folgenden Erweiterung der Problemstellung: Ist es möglich, dass alle Personen in 15 Minuten am gegenüberliegenden Ufer sind und die Königin gesund ist?

Um die soeben skizzierte Variante des BCP zu implementieren, gehen Sie wie folgt vor. Kopieren Sie Ihre Domänenbeschreibungen aus `bcdomain.plan` in eine Datei namens `bcpar.plan` und ermöglichen Sie die Ausführung paralleler Aktionen (Weglassen von `noConcurrency`; sonst keine Modifikationen). Erweitern Sie die Modellierung der Planungsdomäne um die im Folgenden aufgelisteten Aspekte und speichern Sie diese in der Datei `bcswing.plan` ab. (Neu hinzugekommene causal laws in dieser Datei speichern und diese zusammen mit den Dateien `bcpar.plan` und `bctorch.plan` verwenden; siehe auch `dlv` Aufruf unten!):

(p1) Deklarieren Sie ein unäres fluent `rope`(`L`) und eine Aktion `swing`(`P`, `L`), wobei `P` eine Person und `L` jeweils ein Ort (Ufer) ist. Das Fluent repräsentiert den Aufenthaltsort des Seiles, die Aktion dessen Benutzung, um sich an das Zielufer zu schwingen.

(p2) Die Aktion `swing` ist ausführbar, wenn sich das Seil und die ausführende Person am Startufer der Aktion befinden, die Person gesund (`healthy`) ist und gleichzeitig ein Überschreiten der Brücke zum Zielufer stattfindet, wobei die den Schwung ausführende Person nicht gleichzeitig an der Überschreitung beteiligt sein kann.

(p3) In Bezug auf die parallele Ausführung von Aktionen gilt: Es kann nur eine `swing` Aktion parallel mit einer Überschreitung ausgeführt werden. Die parallele Ausführung aller anderen Aktionen ist zu verhindern. Insbesondere können weder mehrere Überschreitungen parallel ausgeführt werden (der Platz auf der Brücke ist nach wie vor beschränkt), noch ist es möglich, dass zwei oder mehr `swing` Aktionen parallel stattfinden (es gibt nur ein Seil). Selbstverständlich sind Überschreitungen weiterhin ohne parallelem `swing` möglich.

(p4) Die Effekte der Aktion `swing` in Bezug auf den Aufenthaltsort des Seils und der ausführenden Person sind offensichtlich; beide befinden sich nach der Ausführung am Zielufer der Aktion. Ist die Fackel zu Beginn der Ausführung nicht bereits am Zielufer, so ist ein weiterer Effekt, dass die ausführende Person nach dem Schwung nicht mehr gesund ($-$`healthy`) ist.

(p5) Der Aufenthaltsort des Seils ändert sich nur durch `swing` Aktionen, d.h., `rope` unterliegt der Inertia.

Testen Sie Ihre Planungsdomäne mit jener Aufgabenstellung, welche aus der vorherigen Problemstellung (`bcproblem2.plan` aus Aufgabe 2.2) hervorgeht, wenn sich das Seil zu Beginn am Ausgangsufer befindet, 15 Minuten zur Verfügung stehen und die Königin am Ende gesund sein soll. Speichern Sie die erweiterte und modifizierte Problemstellung als `bcproblem3.plan` ab.

Verwenden Sie DLV$^{\mathcal{K}}$ mittels

```
dlv -FPopt bcbackground.dl bcpar.plan bctorch.plan
    bcswing.plan bcproblem3.plan -N=60 -planlength=n
```

Bei der Ausgabe eines Planes liefert das System nun einen Plan wie folgt: Aktionen die in einem Schritt parallel ausgeführt werden, sind durch Kommata getrennt, während die einzelnen Schritte durch "`;`" getrennt sind.

Wiederum gilt: Vergessen Sie nicht auf das Testen mit anderen Start- und Zielvorgaben!