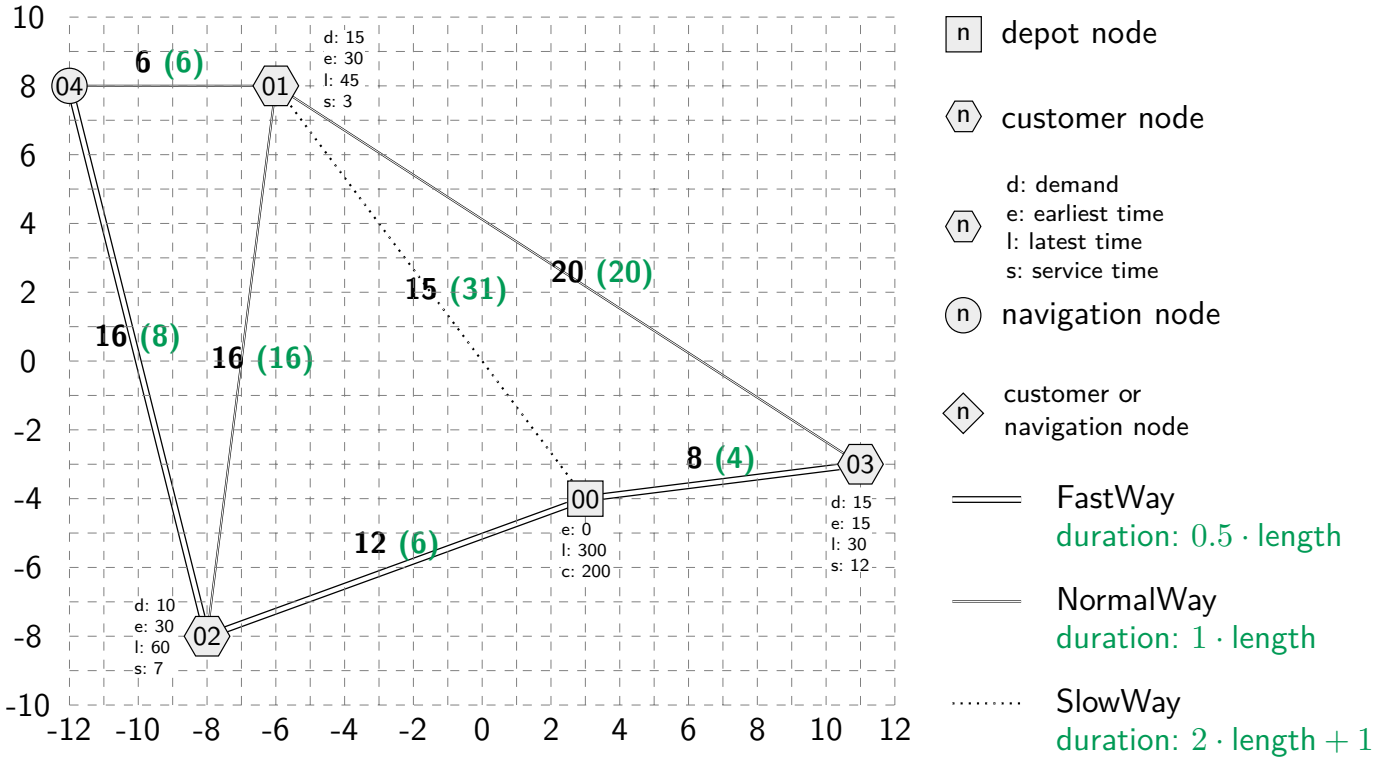
Das zu modellierende Problem



Die Bedarfe, Zeitfenster und Servicezeiten können in der Karte ebenfalls vermerkt werden.

Das Grundgerüst / die Präambel

```
1 model WetterauOrders  ❶
2 products  ❷
3    unit weight 1.0  ❸
```

❶ Jedes Athos-Programm wird mit dem Schlüsselwort `model` eingeleitet, auf das ein (beinahe) frei wählbarer Name für das Modell folgt.

❷ Das Schlüsselwort `products` leitet den Abschnitt ein, indem Produkte definiert werden.

❸ Athos arbeitet (derzeit) eigentlich mit Produkten und zugeordneten Gewichten. Da in diesem Beispiel jedoch bereits Kapazitätseinheiten von der UNitfIX-Software erstellt wurden, führen wir als Produkte eben diese 'units' und ordnen Ihnen ein Gewicht von 1 zu. Damit rechnen wird genau in den Einheiten des Beispiels.

Modellierung der Fahrtzeitfunktionen

```
1 model WetterauOrders
2 products
3    unit weight 1.0
4 functions (1)
5    durationFunction slowWayFunction 2 * length + 1  (2)
6    durationFunction normalWayFunction  length (3)
7    durationFunction fastWayFunction  0.5 * length
```

(1) Das Schlüsselwort `functions` leitet den Abschnitt ein, in dem die Fahrtzeitfunktionen definiert werden.

(2) Jede Fahrtzeitfunktion wird mit dem Schlüsselwort `durationFunction` eingeleitet. Danach erfolgt ein Name für die Funktion sowie ein Ausdruck anhand dessen die Fahrzeit berechnet wird.

(3) Innerhalb eines Ausdrucks zur Berechnung der Fahrtzeit, darf das Schlüsselwort `length` verwendet werden – es bezieht sich dann automatsich auf die jeweilige länger der Straße, der diese Funktion zugeordnet wird.

Modellierung der Depots, Kunden und Navi-Knoten 1

```
1  model WetterauOrders
   :
   :
10 network  ❶
11   nodes  ❷
12     n0 at (-3, -4) isDepot unit  ❸
13       sprouts vehicles
14       customers n1, n2, n3 at 0
15       latestTime 800
16     n1 at (-6, 8)
17       hasDemand unit units 15
18       earliestTime 30 latestTime 45 serviceTime 3
19       // n2, n3 in an analogous way
20     n4 at (-12, 8)
21 }
```

❶ Das `network` Schlüsselwort leitet den Abschnitt zur Definition des Netzwerks ein.

❷ Die Definition der Knoten folgt im Anschluss an das Schlüsselwort `nodes`.

❸ Jeder Knoten benötigt einen Namen (hier: n0), sowie x- und y-Koordinate im Anschluss an das Schlüsselwort `at`. Durch die Angabe `isDepot unit` wird festgelegt, dass von diesem Depot aus Bedarfe an `unit` gedeckt werden.

Modellierung der Depots, Kunden und Navi-Knoten 2

```
1  model WetterauOrders
   :
   :
10 network (1)
11   nodes (2)
12     n0 at (-3, -4) isDepot unit (3)
13       sprouts vehicles (4)
14       customers n1, n2, n3 at 0  (5)
15       latestTime 300 (6)
16     n1 at (-6, 8)
17       hasDemand unit units 15
18       earliestTime 30 latestTime 45 serviceTime 3
19       // n2, n3 in an analogous way
20     n4 at (-12, 8)
21 }
```

(4) Von diesem Depot starten Fahrzeuge vom Typ `vehicles` (wird weiter unten definiert).

(5) Die Kunden n1, n2 und n3 werden von diesem Depot ausgehend beliefert. Die ersten Vehikel starten zum Zeipunkt 0 (der at 0 Teil ist optional und kann auch weggelassen werden).

(6) Bis zum Zeitpunkt 300 müssen alle Fahrzeuge wieder zurück am Depot sein.

Modellierung der Depots, Kunden und Navi-Knoten 3

```
1  model WetterauOrders
   :
   :
10 network (1)
11   nodes (2)
12     n0 at (-3, -4) isDepot unit (3)
13       sprouts vehicles (4)
14       customers n1, n2, n3 at 0  (5)
15       latestTime 300 (6)
16     n1 at (-6, 8)
17       hasDemand unit units 15 (7)
18       earliestTime 30 latestTime 45 serviceTime 3 (8)
19       // n2, n3 in an analogous way
20     n4 at (-12, 8) (9)
```

(7) Durch das hasDemand sowie der Angabe von Produkt (unit) und Menge (units 15) wird ein Knoten zu einem Kunden mit entsprechenden Bedarf.

(8) Hier werden Zeitfenster und Service-Dauer definiert.

(9) Ein einfacher (Navigations-)Knoten.

Modellierung der Kanten und Kanten-Gruppen

- Für das Hinzufügen der eigentlichen Kanten stehen zwei unterschiedliche Herangehensweisen zur Verfügung.
- Die erste Variante eignet sich für Straßen-Typen (Kanten-Typen), von denen es nur sehr wenige Exemplare gibt.
- Die zweite Variante hat zunächst einen gewissen Overhead an Code, spart aber ab einer gewissen Anzahl an Kanten Code-Zeilen ein.

Modellierung der Kanten und Kanten-Gruppen 2

```
1  model WetterauOrders
   :
   :
10 network
   :
   :
20    edges (1)
21       s1 from n0 to n1 function slowWayFunction (2)
22       group normalWayGroup function normalWayFunction members
23          w1 from n1 to n4
24          w2 from n1 to n2
25          w3 from n1 to n3
26       // fastWay group is modelled analogously
```

(1) Straßen bzw. Kanten werden im Anschluss an das Schlüsselwort `edges` definiert.

(2) Hier wird eine einzelne Kante namens `s1` vom Knoten `n0` zum Knoten `n1` definiert. Die Zeit, die ein Fahrzeug zum vollständigen überqueren dieser Kante benötigt, wird mit der `slowWayFunction` berechnet.

Modellierung der Kanten und Kanten-Gruppen 3

```
1  model WetterauOrders
   :
   :
10 network
   :
   :
20    edges (1)
21       s1 from n0 to n1 function slowWayFunction (2)
22       group normalWayGroup function normalWayFunction members (3)
23          w1 from n1 to n4   (4)
24          w2 from n1 to n2   (4)
25          w3 from n1 to n3   (4)
26       // fastWay group is modelled analogously
```

(3) Alternativ kann eine Gruppe von Kanten definiert werden. Dieser Gruppe wird die Fahrtzeitfunktion `normalWayFunction` zugeordnet. Im Anschluss an das `members`-Schlüsselwort werden die Mitglieder der Gruppe definiert. Die Gültigkeit des Schlüsselwortes erstreckt sich bis zur nächsten Gruppe oder bis zum Ende des `edges`-Abschnitts.

(4) `w1`, `w2`, `w3` gehören zur `normalWay`-Gruppe.

Modellierung der Vehikel

```
1  model WetterauOrders
   :
   :
10 products
   :
   :
10 functions
   :
   :
20 network
   :
   :
30 agentTypes (1)
31    agentType vehicles maxWeight 200   (2)
32       behaviour awt awaitTour when finished do die
33       behaviour die vanish
```

(1) Fahrzeuge werden im agentTypes-Abschnitt modelliert, der durch das entsprechende Schlüsselwort eingeleitet wird.

(2) Die Definition eines Vehikel-Typs (Agenten) wird mittels des Schlüsselwortes agentType eingeleitet. es folgt der Name des Typs sowie das maximal zuladbare Gewicht.

(3) In Athos werden Verhaltensweisen für Fahrzeuge definiert. Für statische VRPTWs, wie wir sie kennengelernt haben, ist die mit dem Schlüsselwort behaviour eingeleitete Verhaltensweise stets awaitTour, da das

Modellierung der Vehikel

```
1  model WetterauOrders
   :
   :
10 products
   :
   :
10 functions
   :
   :
20 network
   :
   :
30 agentTypes (1)
31    agentType vehicles maxWeight 200   (2)
32       behaviour awt awaitTour when finished do die (2)
33       behaviour die vanish (2)
```

(4) In Athos werden Verhaltensweisen für Fahrzeuge definiert. Für statische VRPTWs, wie wir sie kennengelernt haben, ist die mit dem Schlüsselwort `behaviour` eingeleitete Verhaltensweise stets `awaitTour`, da das Fahrzeug im Depot wartet, dass es auf Tour geschickt wird. Wenn das Fahrzeug die Tour abgearbeitet hat (`when finished`), soll es das unter `die` definierte Verhalten zeigen.

(5) Das Fahrzeug verschwindet (`vanish`) einfach wieder aus der Simulation.

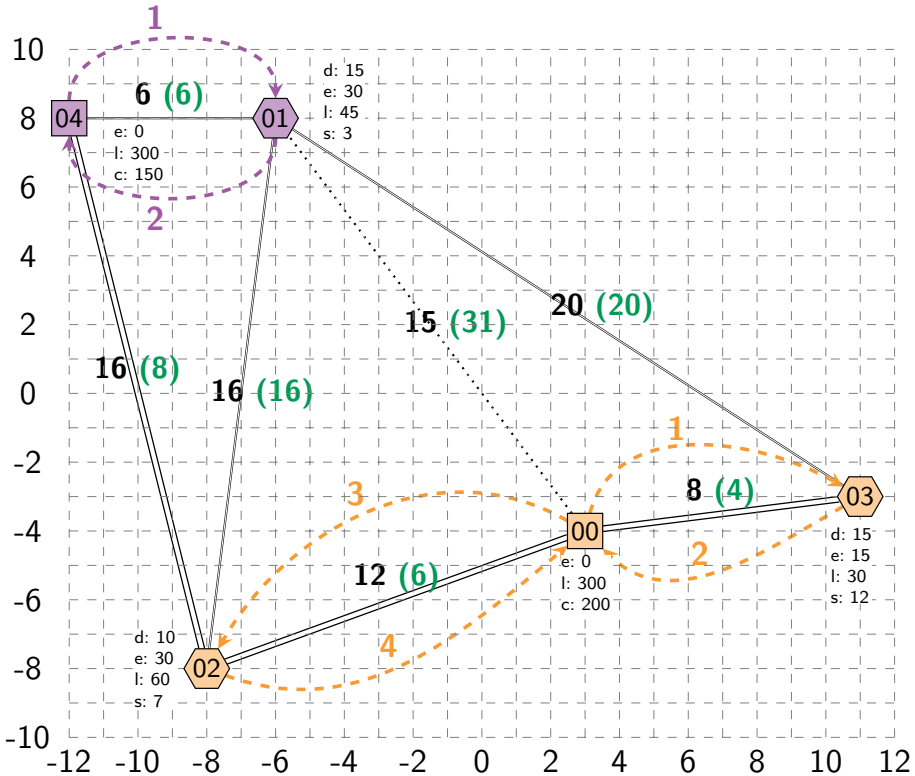
Modellierung der Kennzahlen[1]

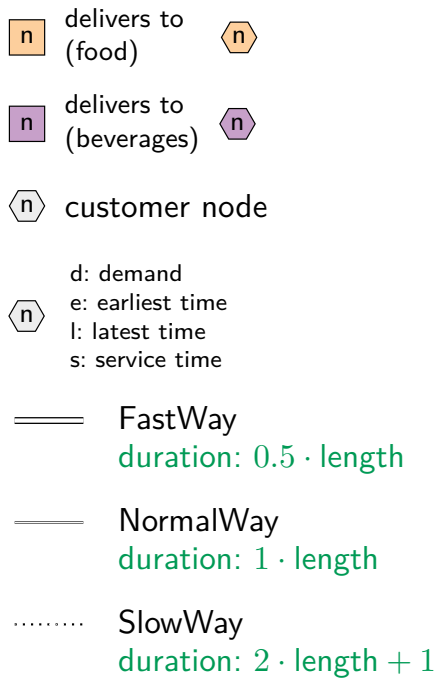
```
1  model WetterauOrders
   :
   :
10 products
   :
   :
10 functions
   :
   :
20 network
   :
   :
30 agentTypes
   :
   :
30 metrics updateRate 10  ❶
31    for vehicles
32       individual metric intendedRoute when notYetSet? set intendedTour
33       class metric distanceCovered when  isAtCustomer?  add distanceTo last customer
34       class metric windowsViolated when isAtCustomer? and latestTime < currentTime add 1
35       class metric windowsMet when isAtCustomer? and currentTime <= latestTime add 1
```

❹ Athos erlaubt die Definition sog. Metriken, mit denen Kennzahlen definiert werden können, die während eines Simulationslaufs berechnet werden sollen. Die `updateRate` setzt das Intervall, in dem die Aktualisierung der Kennzahlen erfolgt.

[1] Wird in den Aufgaben nicht geprüft!

Änderung der Geschäftsstrukturen





Neues Szenario: Getränkemarkt bei Knoten 04, Kunde 01 braucht Getränke. Knoten 00 liefert weiter Lebensmittel, die von den Kunden 02 und 03 bestellt werden.

Änderung der Geschäftsstrukturen

- Viele Elemente des bisherigen Programms können einfach übernommen werden.
  - `Locations` bleiben unverändert.
  - Definition der Kanten und der Fahrtzeiten bleiben identisch.
- Die folgenden Änderungen müssen vorgenommen werden:
  - Die `UNITS = 0` werden nun durch `UNITS_FOOD = 0` und `UNITS_BEVERAGE = 1` ersetzt.
  - `Service.Builder` wird zu `Delivery.Builder` und `Service` zu `Delivery`
  - Der generelle `VehicleType` wird durch zwei spezielle `VehicleTypes` ersetzt, die Kapazitäten für das entsprechende Produkt haben und in der jeweiligen `Location` starten.

Modellierung der Vehikel

```
1  model WetterauOrders2
2  products
3     unitsFood weight 1.0
4     unitsBeverage weight 1.0
   :
   :
10 network
11   nodes
12     n0 at (-3, -4) isDepot unitsFood sprouts vehiclesFood
13       customers n2, n3 at 0 latestTime 300
14     n1 at (-6, 8) hasDemand unitsBeverage units 15
15       earliestTime 30 latestTime 45 serviceTime 3
   :
   :
25     n4 at (-12, 8) isDepot unitsBeverage sprouts vehiclesBeverage
26       customers n1 at 0 latestTime 300
   :
   :
30 agentTypes
31   agentType vehiclesFood maxWeight 200
32     behaviour awt awaitTour when finished do die
33     behaviour die vanish
34   agentType vehiclesFood maxWeight 200
35     behaviour awt awaitTour when finished do die
36     behaviour die vanish
```