Agentensimulation

Agenten

Computergesteuerte Akteure in Spielen (und anderen Simulationen) werden Agenten genannt (manchmal auch MOBs)

Zielsetzung für Agenten ist überzeugend zu sein, nicht zu gewinnen

Möglichkeiten realistischer zu wirken, ohne das grundlegende Verhalten zu verbessern

- Darstellung von Emotionen
- Wiederholungen im Verhalten vermeiden
- Darstellung von zielfremden Verhalten (langsam durch die Gegend bewegen beim Warten, umherschauen, pfeifen, ...)

Agenten steuern

Verschiedene Algorithmen in Verwendung, um Agenten zu steuern (oft Kombinationen)

Da Ziel ist überzeugend zu sein: Fokus auf Algorithmen, die realistisch wirkendes Verhalten liefern, nicht zwingend beste Problemlösefähigkeit

- Zustandsübergangssysteme
- Prioritätstabellen
- Bedürfnishierarchie
- Planungswarteschlange
- Entscheidungslisten
- Entscheidungsbäume
- Verhaltensbäume

Agentenintelligenz

Intelligenz wirkt oft nicht überzeugend

- Erwartungshaltung schlechte künstliche Intelligenz
  $\Rightarrow$ Bei sehr intelligenten Agenten vermuten Spieler eher, dass sie schummeln, als dass sie intelligent sind
- Gute Intelligenz wird oft nur als solche wahrgenommen, wenn sich die Agenten erklären.
  Beispiel: Ein Agent erklärt einem anderen Agenten die Beweggründe für die Handlung „Ich schaue mal hinter dieser Türe nach dem Spieler, weil die Vase kaputt ist“
- Übermenschliche Intelligenz wirkt auch dann nicht menschlich, wenn klar ist, dass nicht geschummelt wird (Schach, Go, Rechenfähigkeiten, . . .)
- Menschliche wirkende Fehler können Darstellung überzeugender machen

Beobachtung durch Spieler

Innerer Zustand eines Agenten, der nicht durch den Spieler erahnbar ist, bringt nichts

$\Rightarrow$ Hinweise darauf geben! Äußerung von Plänen; zeigen von Emotionen; extrem deutliche Körpersprache (undeutliche Körpersprache wird von Spielern als zufälliges Verhalten ignoriert); Hinweise auf Verhaltensaspekt durch andere Akteure, Beispiel: „Ist dir aufgefallen, wie unruhig die Wölfe heute sind?“ kann Spieler motivieren auf den Gemütszustand von Wölfen zu achten und inneren Zustand zu beobachten

Zufall im Verhalten

An vielen Stellen kann Zufall im Verhalten eingebaut werden

- Vorteil: Zufall alleine kann intelligent wirken, innerer Zustand kann vermutet werden, der nicht vorhanden ist, wirkt lebendiger
- Nachteil: Tatsächlicher innerer Zustand kann übersehen werden, bei zu viel Zufall wird vollständiger Zufall vermutet
- Ambivalent: Agenten können durch Spieler nicht mehr so gut vorhergesagt werden

Kombinieren und direkte Verhaltensimplementation

Grundlegende Verhaltensweisen lassen sich gut direkt implementieren

Hauptschwierigkeit in der Auswahl der Verhaltensweisen

Viele der genannten Algorithmen erlauben andere Algorithmen als Bestandteile, damit gute Kombinierbarkeit

Zustandsübergangssysteme

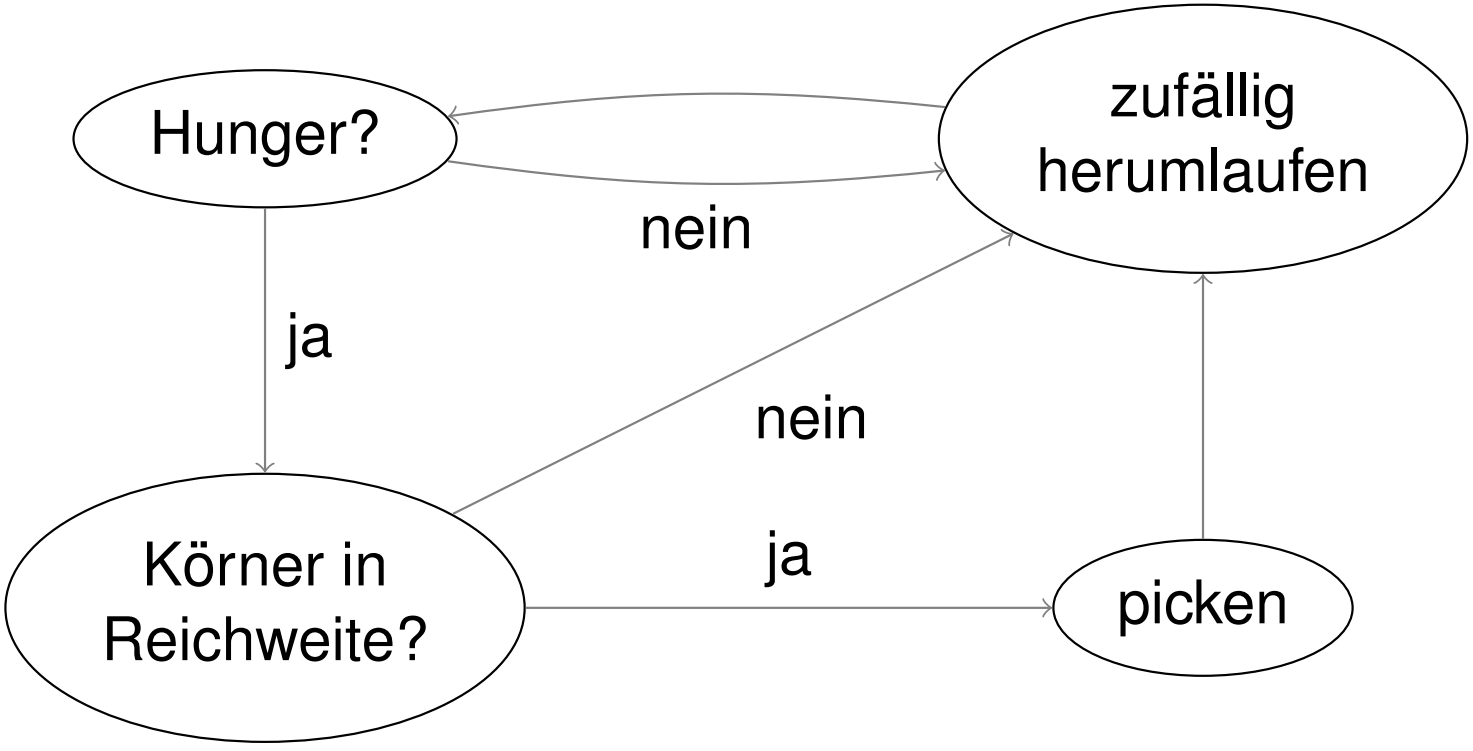
Zustandsübergangssysteme bekannt aus Kapitel „zusammenfassende Berechnungen“

Mit Aktionen (Hooks, Codereferenzen) an den Zuständen: Nahezu beliebiges Verhalten darstellbar

- Einfache Verhaltensweisen sehr einfach umsetzbar
- Gute Maschinenlernverfahren für Markovketten bekannt
- Für kompliziertere Verhaltensweisen sind andere Modelle geeigneter: Zustandsraum kann extrem groß werden
- Zustandsübergangssysteme als Aktionen an Zustandsübergangssystemen: Bessere Übersichtlichkeit, weniger Zustände notwendig; bekannt als hierarchische Zustandsübergangssysteme

Beispiel: Zustandsübergangssystem

Bewegungssteuerung eines Huhns



Prioritätstabellen

Für verschiedene Verhaltensweisen werden verschiedene Prioritäten berechnet

Verhalten mit der höchsten Priorität wird ausgeführt

Beispiel: Hahn

- Hungerwert 0-1 gibt Prioritätswert 0-5 für Aktion Körner suchen
- Fuchs in Nähe gibt Priorität 8 für Stall aufsuchen
- Müdigkeitswert 0-1 gibt Priorität 0-3 für Stall aufsuchen
- Nach 5:00, vor 8:00 und heute noch nicht gekräht gibt Prioritätswert 2 zum Krähen

Bedürfnishierarchie

Prioritätsreihenfolge von Bedürfnissen: Nur wenn die darunter liegenden Bedürfnisse erfüllt sind, werden die darüberliegenden berücksichtigt, immer das unterste unerfüllte wird versucht zu erreichen

Beispiel:

- Soziale Interaktion
- Schlaf
- Hunger
- Durst
- Körperliche Unversehrtheit (Abstand von Gegnern, . . .)

Planungswarteschlange

Warteschlange, in der Pläne stehen.

Plan am Anfang der Schlange wird bearbeitet, kann in Teilpläne zerteilt werden (Situationsabhängig)

Dringende Pläne können vorne angestellt werden (Gegner in Sicht), weniger dringende (Wohnung reinigen) in der Mitte oder hinten

Pläne können umsortiert werden, damit zueinander passende Pläne zeitlich nahe ausgeführt werden (räumlich nahe beieinander; beim bearbeiten eines neuen Plans mehrere am Anfang anschauen, ob einer an diesem Ort bearbeitbar ist; . . .)

Entscheidungslisten

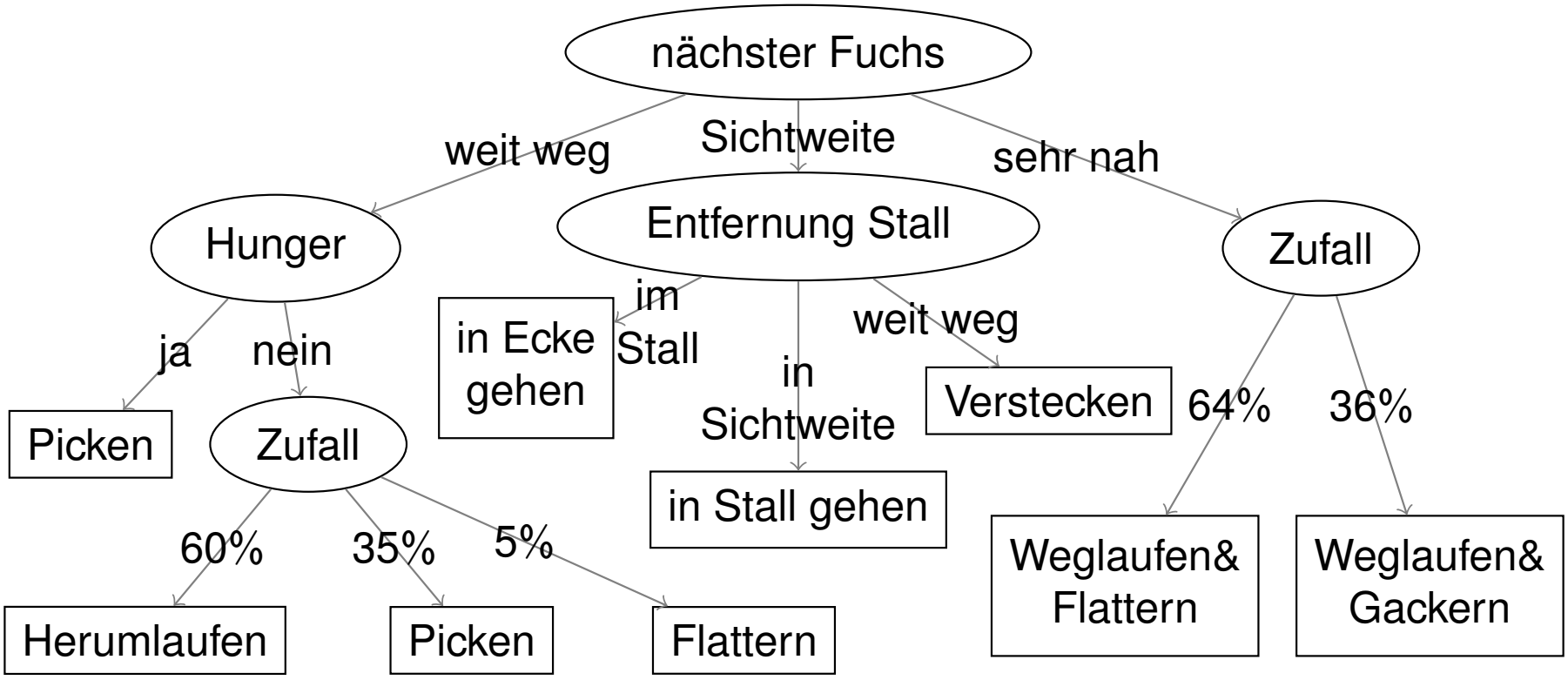
Liste von Bedingungen und Aktionen; erste Aktion mit erfüllter Bedingung wird ausgeführt

Problem 1: Lange Liste muss dauernd durchgearbeitet werden, ineffizient

Problem 2: Längere Listen unübersichtlich

Entscheidungsbäume

Effizientere Version der Entscheidungslisten:
Baum mit Enscheidungsfragen an inneren Knoten, unterschiedlichen Kindern für die Antworten und Verhaltensweisen an den Blättern



Entscheidungsbäume, Eigenschaften

Kleine Entscheidungsbäume können bereits komplexes Verhalten modellieren

Entscheidungsbäume gut maschinell lernbar (ohne Zufall, dafür mehrere lernen und zufällig einen verwenden)

Baum als Modell ausreichend: Aktionen können wieder Entscheidungsbäume sein

Nachteile: System rein reaktiv, innerer Zustand nicht direkt in der Modellierung (durch Aktionen, die inneren Zustand setzen machbar, schlecht für Übersichtlichkeit)

Verhaltensbäume

Seit etwa 2000 bekannt, popularisiert durch Halo 2, Spore und Bioshock, sowie die Unreal Engine.
Verbreitet in Computerspiele- und Robotikindustrie
Erweiterung von Entscheidungsbäumen: Gibt Kombinatoren für Verhalten, sowie zeitliche Komponenten

- Gibt die Konstrukture von Entscheidungsbäumen: Aktionen und Entscheidungsknoten
- Jeder Aktionsknoten gibt zusätzlich an, ob die Aktion ein Erfolg oder ein Fehlschlag war
- Weitere Sorte innerer Knoten: Dekoratoren.
  Hat genau ein Kind, bearbeitet das Ergebnis des Kinds
- Weitere Sorte innerer Knoten: Kombinatorknoten.
  Hat Liste von Kinder, kombiniert ihre Verhaltensweisen in der Ausführung

Verhaltensbäume nicht vollständig einheitlich in Verwendung, gibt verschiedene Erweiterungen und Einschränkungen

Verhaltensbäume ausführen

Ergebnis eines Verhaltensbaums zu berechnen nennt man von Ausführen.

Ergebnis ist „Erfolg“, „Fehlschlag“, oder „in Bearbeitung“; „in Bearbeitung“ hat zusätzliche Werte, den sogenannten Ausführungszustand

Aktionen haben Codereferenz, diese kann alle 3 Rückgabewerte erzeugen, kann ihren internen Ausführungszustand fortsetzen

Gibt ein Kind „in Bearbeitung“ zurück, so passiert einheitlich das Folgende in fast allen Knoten (außer Parallel, spätere Folie)

- Knoten selbst gibt ebenfalls „in Bearbeitung“ zurück
- Ausführungszustand ist innerer Zustand des Knotens, sowie Information, welches Kind gerade ausgeführt wird, sowie Ausführungszustand des Kindes in Ausführung

Verhaltensbäume, Dekoratoren

Jeder Dekorator hat genau ein Kind

- Inverter: Kind wird ausgeführt, Ergebnis invertiert: Bei Erfolg Fehlschlag zurückgeben, bei Fehlschlag Erfolg
- Warteknoten: Ausführung wird an der Stelle für eine feste Zeitdauer angehalten, danach wird das Kind ausgeführt
  Um das zu tun, gibt der Warteknoten „in Bearbeitung“ zurück; wird er nach der Wartezeit fortgesetzt, wird das Kind ausgeführt
- Feste Schleife: Kinderknoten wird mehrmals ausgeführt, festgelegte Anzahl
- While-Schleife: Kinderknoten wird solange wiederholt, bis er Fehlschlag zurückgibt; gibt Fehlschlag zurück, wenn Kind nie erfolgreich war
- Retry-Schleife: Kinderknoten wird solange wiederholt, bis er Erfolg zurückgibt; dann Erfolg zurückgeben
- Debug: Kind wird normal ausgeführt, aber es wird eine Debugnachricht ausgegeben, wenn die Bearbeitung anfängt und wenn sie zuende ist

Verhaltensbäume, Kombinatoren

Jeder Kombinator hat eine Liste an Kindern; kann fest oder randomisiert sein

- Sequenz:
  Alle Kinder werden der Reihe nach ausgeführt, es sei denn, ein Kind gibt Fehlschlag zurück, dann Fehlschlag zurückgeben.
  Waren alle Kinder erfolgreich, dann Erfolg zurückgeben
- Selektor (auch Fallback genannt):
  Wie Sequenz, nur Rolle von Erfolg und Fehlschlag vertauscht.
- Parallel:
  Alle Kinder werden „gleichzeitig“ ausgeführt, d.h. die Kinder werden in Reihenfolge ausgeführt, gibt aber ein Kind „in Bearbeitung“ zurück, so werden die anderen Kinder auch ausgeführt.
  Sobald ein Kind „Erfolg“ oder „Fehlschlag“ zurückgibt, wird das vom Parallel-Knoten auch zurückgegeben

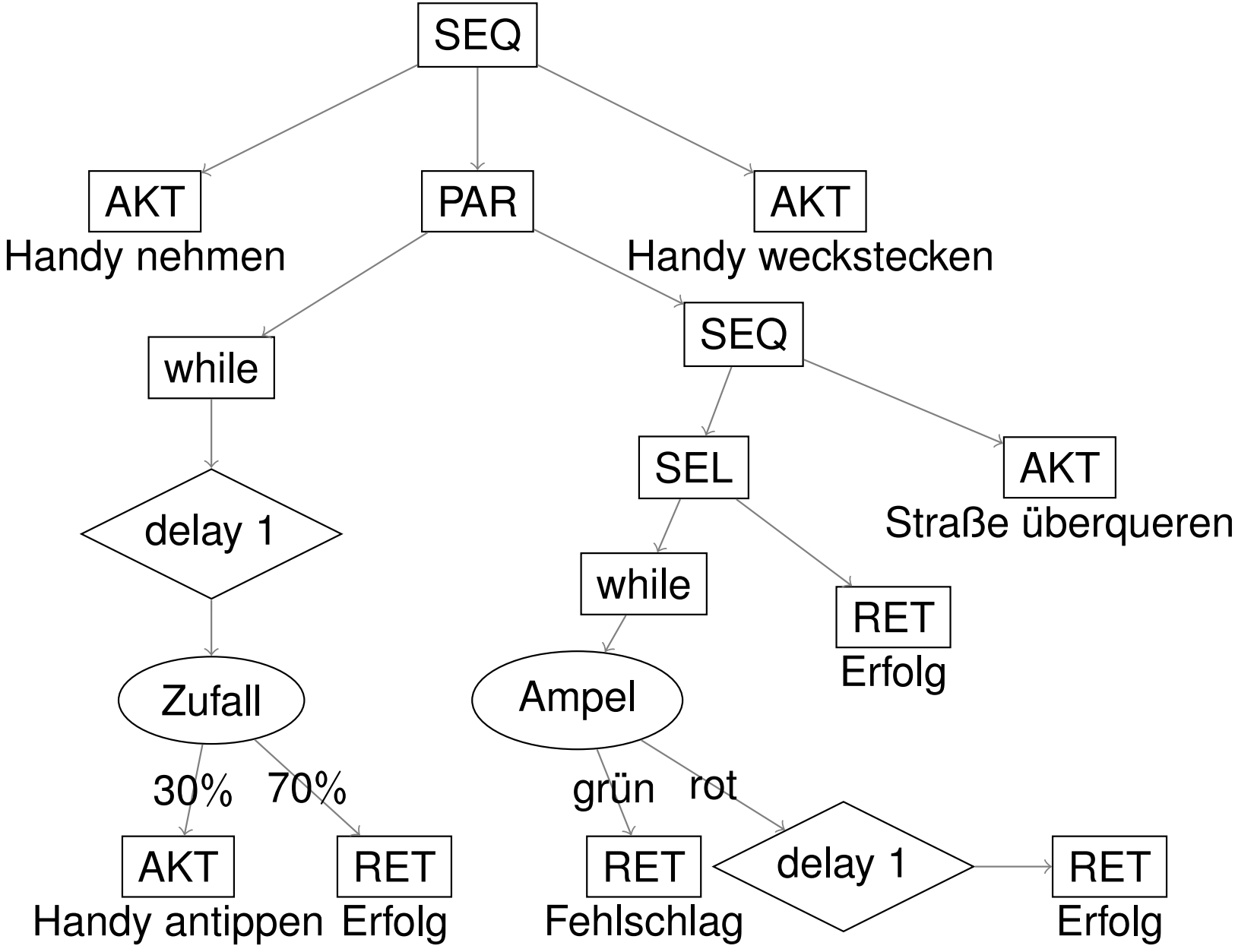
Verhaltensbäume, Parallelkombinator

Es gibt auch Parallelkombinatoren mit komplexeren Regeln, wie mit Rückgaben umzugehen ist

Zustandscounter in Aktionen innerhalb eines Parallelkombinators können Semaphoren simulieren

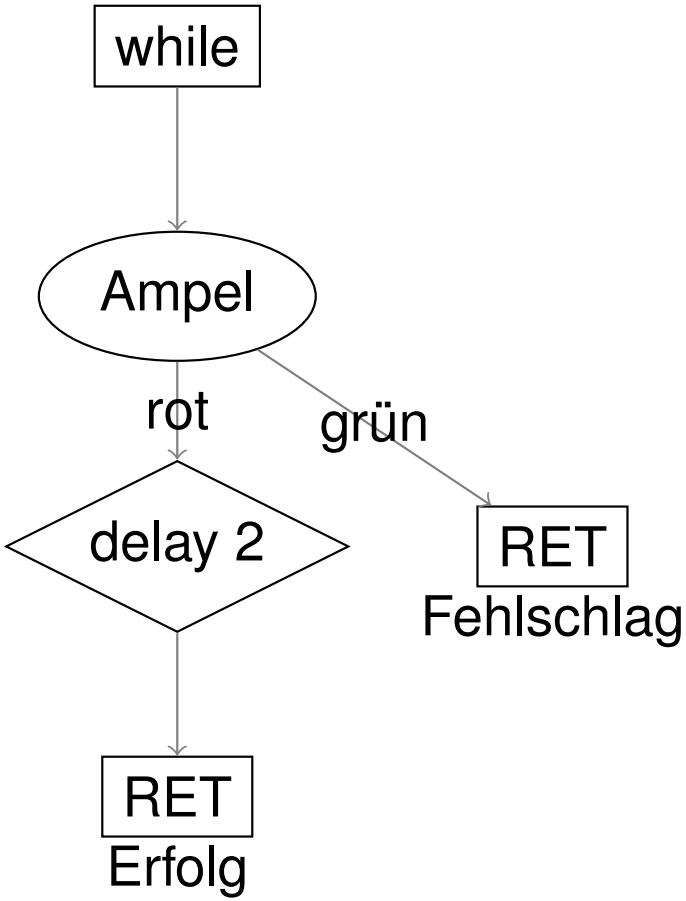
Parallelkombinatoren mit Bedacht einsetzen: Können schnell unübersichtlich werden

Beispiel Verhaltensbaum: Ampelüberquerung



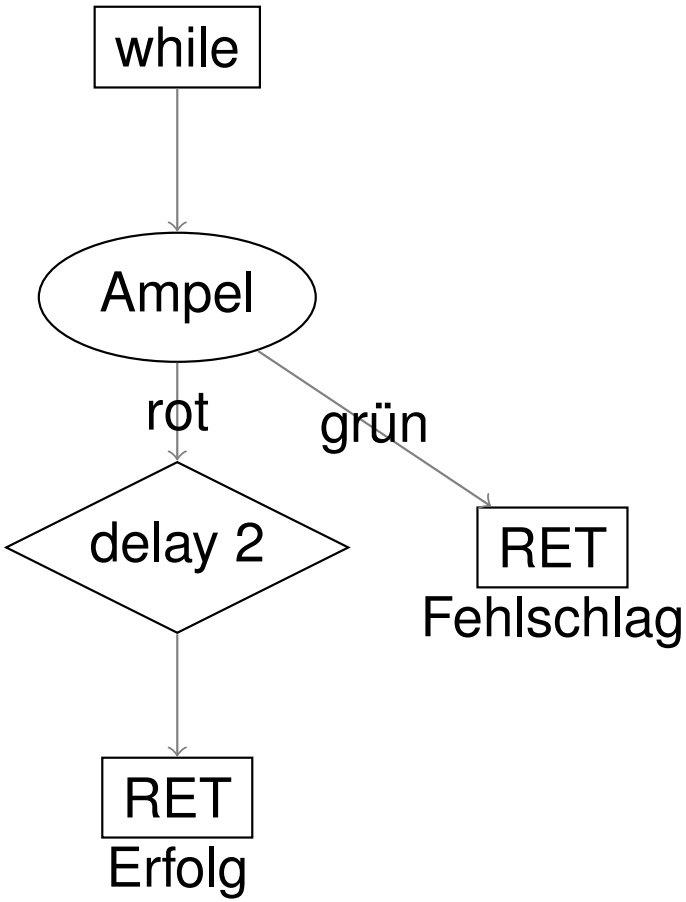
Beispiel Ausführung Verhaltensbaum

Beispielaufruf gekürzter Verhaltensbaum, Startzeitpunkt: 343



Beispiel Ausführung Verhaltensbaum

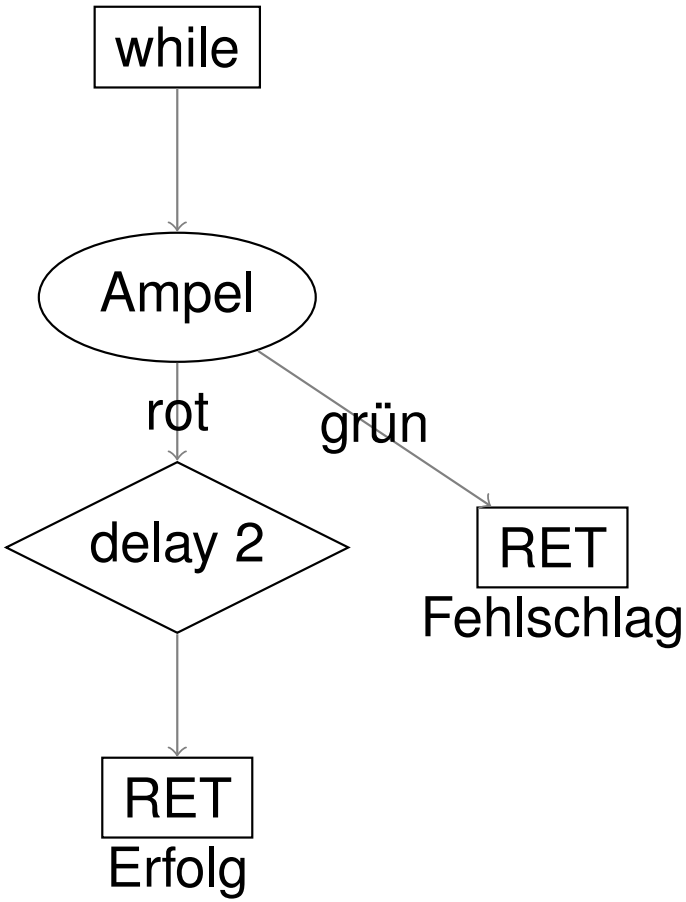
Beispielaufruf gekürzter Verhaltensbaum, Startzeitpunkt: 343



t=343 Kein Aufrufzustand aus altem Aufruf, fängt mit while-Knoten an; Rückgabe (while (Ampel rot (delay 345)))

Beispiel Ausführung Verhaltensbaum

Beispielaufruf gekürzter Verhaltensbaum, Startzeitpunkt: 343

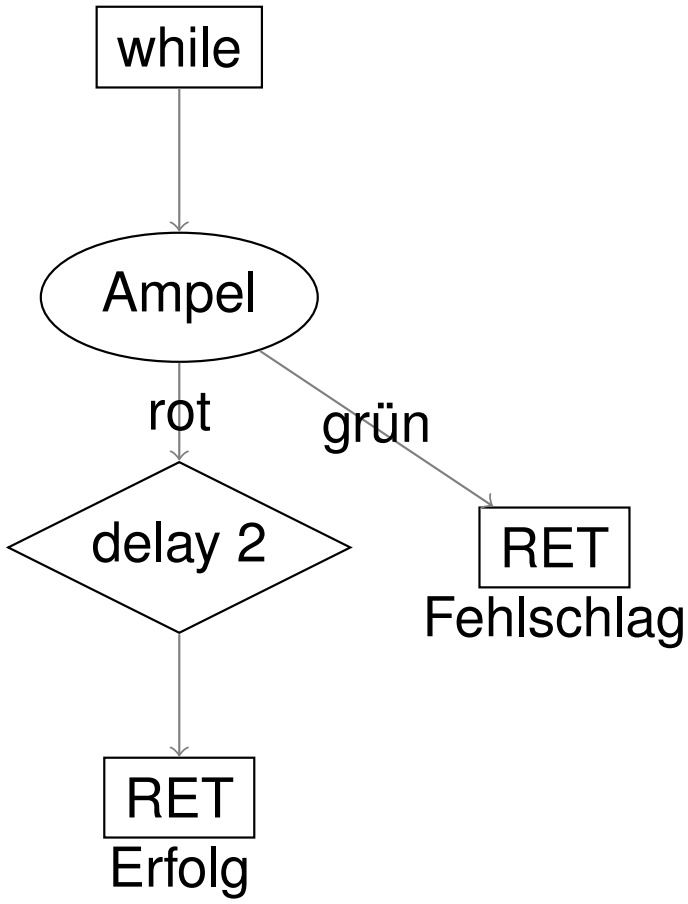


t=343 Kein Aufrufzustand aus altem Aufruf, fängt mit while-Knoten an; Rückgabe (while (Ampel rot (delay 345)))

t=344 Delay ist noch nicht bereit, darum gleicher Rückgabewert

Beispiel Ausführung Verhaltensbaum

Beispielaufruf gekürzter Verhaltensbaum, Startzeitpunkt: 343



t=343 Kein Aufrufzustand aus altem Aufruf, fängt mit while-Knoten an; Rückgabe (while (Ampel rot (delay 345)))

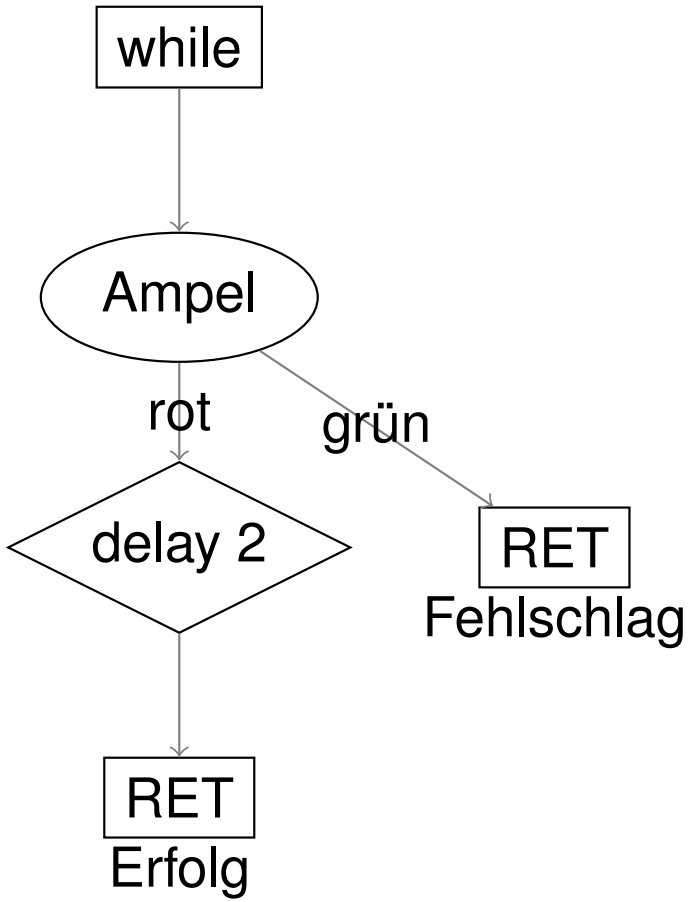
t=344 Delay ist noch nicht bereit, darum gleicher Rückgabewert

t=345 Delay führt Kind aus, gibt Erfolg zurück. Damit gibt Ampel Erfolg zurück. Darum startet while Ampel neu, Rückgabewert (while+ (Ampel rot (delay 347)))

while+: Kennzeichnug, dass nicht erster while-Aufruf

Beispiel Ausführung Verhaltensbaum

Beispielaufruf gekürzter Verhaltensbaum, Startzeitpunkt: 343



t=343 Kein Aufrufzustand aus altem Aufruf, fängt mit while-Knoten an; Rückgabe (while (Ampel rot (delay 345)))

t=344 Delay ist noch nicht bereit, darum gleicher Rückgabewert

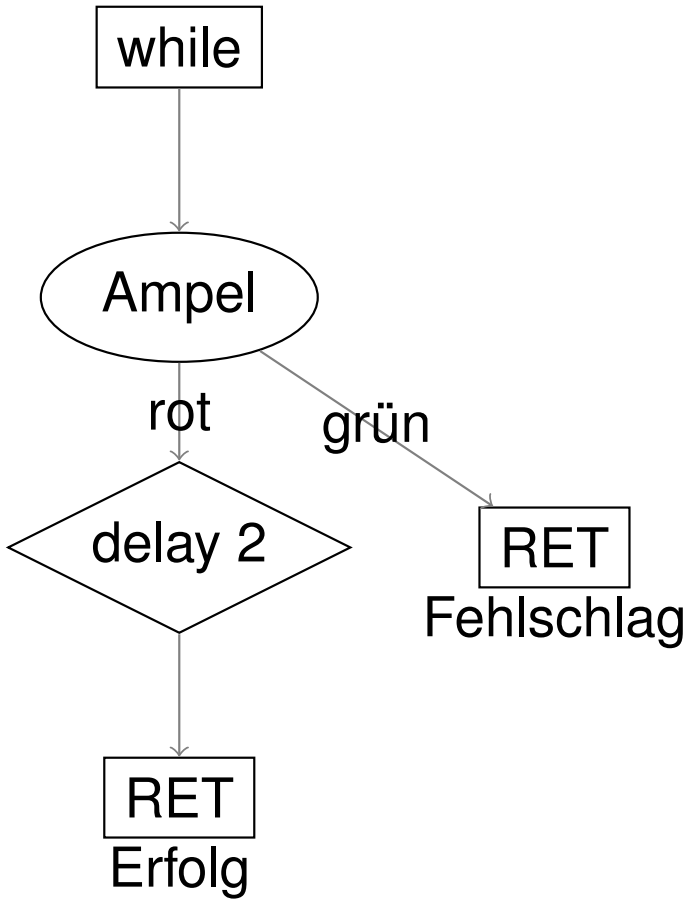
t=345 Delay führt Kind aus, gibt Erfolg zurück. Damit gibt Ampel Erfolg zurück. Darum startet while Ampel neu, Rückgabewert (while+ (Ampel rot (delay 347)))

t=346 Delay ist noch nicht bereit, darum gleicher Rückgabewert

while+: Kennzeichnug, dass nicht erster while-Aufruf

Beispiel Ausführung Verhaltensbaum

Beispielaufruf gekürzter Verhaltensbaum, Startzeitpunkt: 343



t=343 Kein Aufrufzustand aus altem Aufruf, fängt mit while-Knoten an; Rückgabe (while (Ampel rot (delay 345)))

t=344 Delay ist noch nicht bereit, darum gleicher Rückgabewert

t=345 Delay führt Kind aus, gibt Erfolg zurück. Damit gibt Ampel Erfolg zurück. Darum startet while Ampel neu, Rückgabewert (while+ (Ampel rot (delay 347)))

t=346 Delay ist noch nicht bereit, darum gleicher Rückgabewert

t=347 Delay führt Kind aus, gibt Erfolg zurück. Damit gibt Ampel Erfolg zurück. Diesmal Ampel grün, Fehlschlag beendet while, while wurde aber mindestens einmal ausgeführt, Rückgabe Erfolg

while+: Kennzeichnug, dass nicht erster while-Aufruf