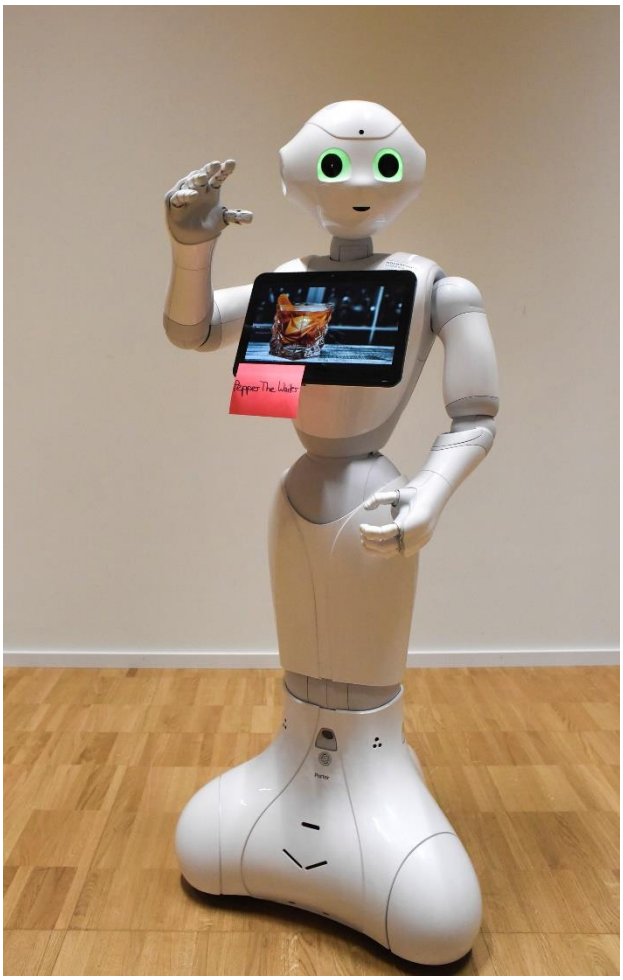
HSLU Hochschule Luzern
B.Sc. Informatik

Pepper the waiter

ROBLAB-HS20 / Team 2



Projektbetreuung: Dr. Florian Herzog

Abgabetermin: 16.12.2020

Projektteam:

Willi Adrian adrian.willi@stud.hslu.ch
Hostettler Maurizio maurizio.hostettler@stud.hslu.ch

Inhaltsverzeichnis

1 Geplantes Verhalten 3

2 PEAS Beschreibung 3

2.1 Annahme zur Umgebung/Eigenschaften der Umgebung 3

2.2 Eingesetzte Aktuatoren/Sensoren 4

2.3 Eingesetzte externe Software 4

2.4 Performance Measure 4

3 Lösungsansatz 5

3.1 Agententypen – Model-Based Reflex Agent 5

3.2 Agentenarchitektur 5

3.3 Übersichtsdiagramm zur Lösung 5

3.4 Implementierung des Verhaltens 6

3.5 Human-Robot Collaboration 6

4 Evaluation der Lösung 6

4.1 Was wurde erreicht 6

4.2 Grenzen des Verhaltens 6

4.3 Flexibilität und Robustheit des Agenten 7

4.4 Risiken und Unsicherheiten des programmierten Verhaltens 7

4.5 Verbesserungspotenzial 7

5 Reflexion 7

6 Anhang 8

6.1 Finite State Machine 8

Abbildungsverzeichnis

Abbildung 1 - Ablauf des Verhaltens 3
Abbildung 2 - Finite State Machine (Maximierte Version im Anhang) 5
Abbildung 3 - Grobübersicht Software 6

1 Geplantes Verhalten

Das Ziel des vorgesehenen Verhaltens ist, dass der Roboter selbstständig einen Kunden in einer Bar findet, die Bestellung aufnimmt und anschliessend das Getränk dem korrekten Kunden bringt. Im geplanten Verhalten trägt der Roboter aber kein echtes Getränk, da weder die Hardware noch die Software vom Pepper dafür geeignet ist. Somit liegt der Fokus einerseits auf der korrekten Navigation im Raum und andererseits in der Kommunikation mit dem Kunden. Gerade hier soll der Roboter in der Lage sein, unter anderem auf Fragen des Kunden bezüglich beispielsweise den verfügbaren Getränken antworten zu können. Im untenstehenden Automaten wurde der Ablauf mit Zuständen und Zustandsübergängen modelliert. Zusätzlich wurden die verschiedenen Tätigkeiten grob in drei Kategorien (Detection, Navigation, Communication) unterteilt. «Pepper the waiter» muss also in all diesen Teilgebieten Fähigkeiten haben, um das geplante Verhalten zu erreichen.

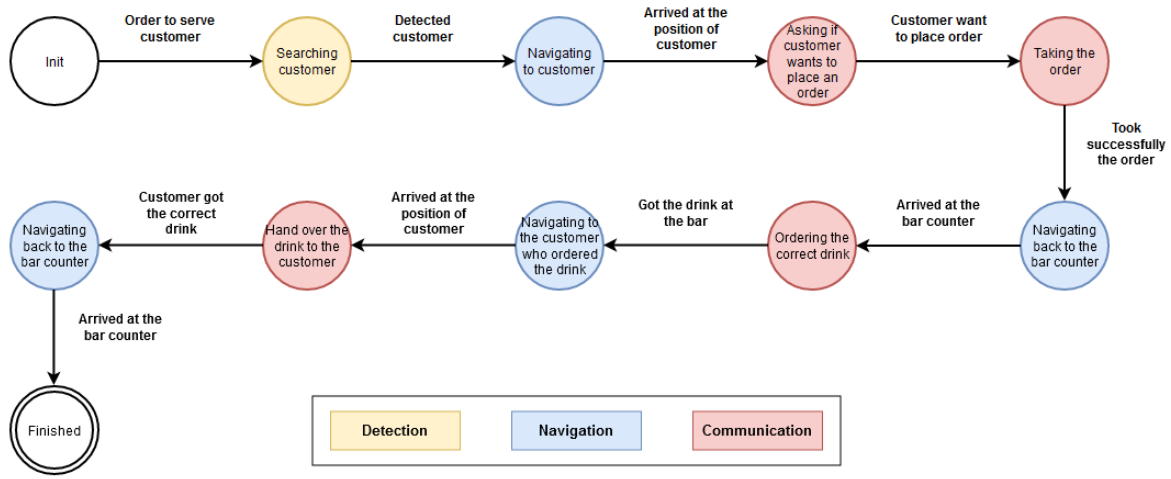


Abbildung 1 - Ablauf des Verhaltens

2 PEAS Beschreibung

2.1 Annahme zur Umgebung/Eigenschaften der Umgebung

Die Umgebung spielt beim hier gewählten Verhalten eine sehr zentrale Rolle und hat einen grossen Einfluss auf die Schwierigkeit. Die Eigenschaften der Umgebung sind ein fixer Raum beziehungsweise konkret die Bar, mehrere Personen, wobei eine Person die Funktion als Barkeeper hat sowie mindestens noch ein Kunde. Für den Roboter ist vor allem das Auffinden der Kunden sowie des Barkeepers relevant. Die Umgebung kann zudem kontrollierter gestaltet werden, indem sich beispielsweise nur zwei Kunden in der Bar aufhalten und keine zusätzlichen Hindernisse vorhanden sind. Beide Massnahmen wurden in der vorliegenden Arbeit umgesetzt. Nachfolgend ist die Beurteilung der Umgebung ersichtlich. Zudem bleibt der Kunde am selben Platz stehen, während der Roboter das Getränk an der Bar bestellt. Weiter kann der Kunde Englisch sprechen.

Observability	**Partially observable** – Die Sensoren decken nur einen Teilbereich ab und können nicht die ganze Bar erfassen
Predictability	**Stochastic** – Ein Kunde kann unerwartet reagieren
Time dependency	**Sequential** – Pepper muss das Getränk der richtigen Person bringen
Dynamics	**Dynamic** – Neue Kunden können die Bar betreten
Discrete vs. Continuous	**Continuous** – Es können unendlich viele Möglichkeiten eintreten
Number of agents	**Model-based reflex agent** – Pepper ist mit einer teilweise sichtbaren Umgebung konfrontiert, hat ein internes Weltmodell und merkt sich, wo in diesem Modell er sich befindet

2.2 Eingesetzte Aktuatoren/Sensoren

Damit das beschriebene Verhalten umgesetzt werden kann, werden verschiedene Sensoren und Aktuatoren benötigt. Im Nachfolgenden werden diese kurz aufgeführt und deren Haupteinsatzgebiet im geplanten Verhalten angegeben.

Sensoren

- **2D Kamera –** Gesicht der Kunden merken, Kunden in der Bar finden
- **Mikrofone –** Spracherkennung bei der Kommunikation, Ortung von Geräuschquellen
- **Sonar –** Abstände messen für ein sicheres Navigieren in der Bar

Aktuatoren

- **Sprachausgabe –** Kommunikation mit Kunden und Barkeeper
- **Bewegung –** Navigation innerhalb der Bar zum Kunden und zum Bartresen
- **Gesten (Körper/Arme/Kopf) –** Gesten für die nonverbale Kommunikation mit dem Kunden

2.3 Eingesetzte externe Software

Die in NAOqi integrierte Erkennung von Personen hat nicht zu den erwünschten Resultaten geführt. Speziell wenn das Gesicht nicht sichtbar war, traten Schwierigkeiten in der Erkennung von Personen auf. Aus diesem Grund wurde auf die Cloud Vision API[1] von Google zurückgegriffen. Nun werden die Bilder für die Analyse an diesen Service gesendet, der dann sehr schnell die gefunden Merkmale zurückmeldet. Der Service funktioniert zuverlässig und der Roboter kann dadurch sehr gut Personen im Raum erkennen, die beispielsweise auch nicht gerade zu ihm stehen.

2.4 Performance Measure

Die Kriterien für die Performance Measure wurden gemäss der untenstehenden Tabelle festgelegt. Wichtige Aktionen wurden mit einer höheren Punktzahl belegt. Dies wäre dann vor allem bei einer weiteren Entwicklung sehr hilfreich, da sich so ein realistischeres Bild zeigt.

Event	Punkte
Ein Gast bekommt nach der Bestellung den richtigen Drink	+ 250
Pepper kann eine Empfehlung abgeben (Bonus)	+ 50
Pepper findet initial keinen Gast, obwohl einer da ist	- 100
Pepper muss nachfragen, weil er etwas nicht verstanden hat	- 50
Pepper findet den Weg zurück zur Bar nicht	- 50
Total ohne Bonus und Abzüge	250

Tabelle 1 - Performance Measure

[1] https://cloud.google.com/vision

3 Lösungsansatz

3.1 Agententypen – Model-Based Reflex Agent

Für die Umsetzung des gewählten Verhaltens wird ein Model-Based Reflex Agent verwendet. Wie unter PEAS bereits erwähnt wurde, befindet sich Pepper bei der Bar in einer Umgebung, die er nicht komplett überschauen und kennen kann. Um seine Aufgabe wahrzunehmen muss, er sich trotzdem in dieser Umgebung bewegen und orientieren können. Dabei hilft ihm ein einfaches internes Weltmodell. Zudem hat der Roboter interne Zustände, die von der vergangenen sowie aktuellen Wahrnehmung der Umgebung abhängen. Wenn Pepper beispielsweise an der Bar ein Getränk bestellt, muss er wissen, welches Getränk der Kunde gewünscht hat. Damit der Model-Based Reflex Agent in einen neuen Zustand kommt, muss er Informationen über die Entwicklung seiner Umgebung sowie über die Folgen auf die direkte Umgebung einer von ihm ausgeführten Aktion haben.

3.2 Agentenarchitektur

Die Einordnung von «Pepper the waiter» bezüglich einer Agentenarchitektur ist gar nicht so einfach, da nicht von Beginn an auf eine explizite Architektur gesetzt wurde. Dennoch ist «Pepper the waiter», wenn auch viel jünger, dem Shakey (deliberative System) von 1983 ähnlich. Die Autoren sehen aber auch Elemente aus einem reaktiven System (reactive System) in «Pepper the waiter».

Dem Roboter wird eine Aufgabe gestellt und um diese umzusetzen, müssen verschiedene Status durchlaufen werden. Dabei wird einerseits die Sense-Plan-Act Methode der überlegten Systeme verwendet, um zu einem Zwischenziel und somit zum nächsten Status zu gelangen. Ein Beispiel dafür ist die Erkennung einer Person im Raum. Zuerst wird mit der Kamera ein Bild gemacht (Sense), anschliessend wird die zu fahrende Distanz berechnet (Plan) und schlussendlich findet die tatsächliche Bewegung zur Person hin statt (Act). Andererseits werden bei den Gesprächen typische Elemente reaktiver Systeme verwendet. Pepper wartet dabei auf einen Input (Sense) und handelt basierend auf dem Input direkt, indem er beispielsweise eine Antwort gibt (Act).

3.3 Übersichtsdiagramm zur Lösung

Nachfolgend ist die gewählte Lösung als eine Finite State Machine dargestellt. Die Grafik ist ebenfalls im Anhang in einer vergrösserten Form verfügbar. Der Roboter befindet sich jeweils in einem Zustand und wechselt diesen, wenn ein entsprechender Input kommt. Zugrunde liegt hierbei der oben beschriebene Model-Based Reflex Agent. Wenn das Verhalten wie vorgesehen abgearbeitet werden kann, dann durchläuft der Roboter nacheinander die Zustände S1 bis S11. Tritt etwas Unvorhergesehenes auf, kann der Roboter verschieden reagieren und etwa Zustände überspringen. Beispielsweise wenn ein Kunde kein Getränk bestellen möchte.

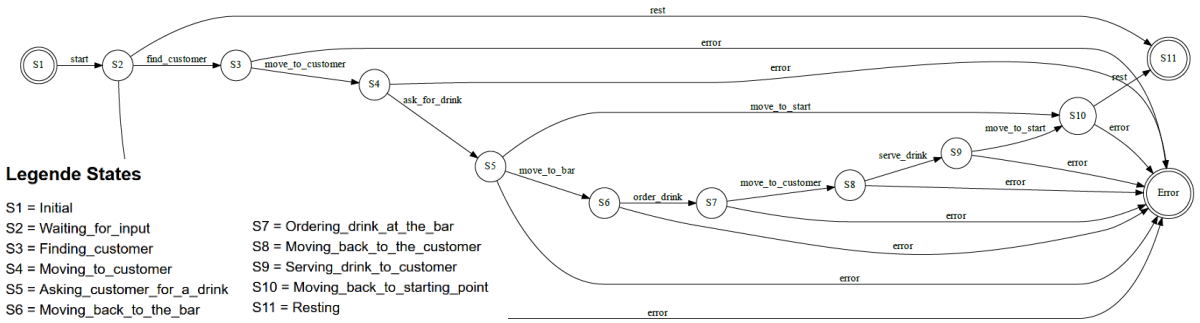


Abbildung 2 - Finite State Machine (Maximierte Version im Anhang)

3.4 Implementierung des Verhaltens

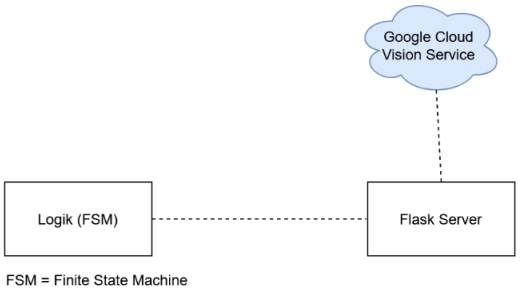


Abbildung 3 - Grobübersicht Software

Das Verhalten wurde in kleinere Einheiten aufgeteilt und dann umgesetzt. Um die Verantwortlichkeiten im Team einfach zu halten und effizient arbeiten zu können, wurden die Zuständigkeiten über die kleineren Einheiten im Team aufgeteilt. Die kleineren Komponenten wurden dann laufend mit Hilfe einer Finite State Machine integriert und getestet. Der Service, der die Bilder an die CloudVisionAPI sendet, wurde vom eigentlichen Code getrennt, da es unter anderem Probleme mit den Versionen von Python gab. Dieser interne Service wird als Flask-Server zur Verfügung gestellt und kann einfach angesprochen werden.

3.5 Human-Robot Collaboration

Gerade im vorliegenden Verhalten spielt die Human-Robot Collaboration eine wichtige Rolle, da «Pepper the waiter» viel Kontakt mit Personen hat. Daher wurden verschiedene Massnahmen getroffen, um die Erfahrung mit dem Roboter angenehmer zu machen. Damit Pepper freundlicher und weniger wie eine Maschine daherkommt, gestikuliert er während den Interaktionen mit den Personen. Zudem kommuniziert er auf eine freundliche und angenehme Art. Weiter achtet Pepper bei der Navigation zum Kunden darauf, nicht näher als auf einen halben Meter hin zu fahren und so ausserhalb der intimen Zone zu bleiben und dennoch nahe genug zu sein, um ein Getränk zu servieren.

4 Evaluation der Lösung

4.1 Was wurde erreicht

«Pepper the waiter» kann das vorgesehene Verhalten zuverlässig erfüllen. Der Roboter kann Personen im Raum erkennen und zu diesen navigieren. Anschliessend ist er in der Lage, eine Bestellung aufzunehmen und dabei unter anderem eine Empfehlung abzugeben, die Getränke am Bartresen zu holen und wieder zum Kunden zu bringen. Wenn sich die Umgebung wie angenommen verhält, dann funktioniert das Verhalten mit wenigen Ausnahmen in allen Fällen gut, wobei hier die Performance Measure Kriterien zum Tragen kommen für die Bewertung.

4.2 Grenzen des Verhaltens

Die Anwendung der Performance Measure Kriterien auf alle Probeläufe hebt aber auch die Grenzen hervor. So hat Pepper manchmal Probleme, den Kunden beim ersten Mal zu verstehen und muss nachfragen. Ein Beispiel dazu ist am Schluss des Videos zu sehen. Seltener hatte der Roboter Schwierigkeiten bei der Navigation und konnte entweder nicht genau zum Kunden oder zurück zur Bar navigieren. Zudem würde eine nicht angenommene Änderung (Bsp. der Kunde geht nach der Bestellung an einen anderen Ort in der Bar) der Umgebung dazu führen, dass das Verhalten an seine Grenzen stösst. Dies wäre etwa der Fall, wenn sich der Kunde, der ein Getränk bestellt hat, von seinem Standort entfernen würde. Pepper wäre dann nicht mehr in der Lage, ihn zu finden. Aus diesem Grund wurde dieser Fall bei den Annahmen zur Umgebung ausgeschlossen.

4.3 Flexibilität und Robustheit des Agenten

Innerhalb des vorhergesehenen Verhaltens sowie den darin möglichen Optionen ist die Flexibilität und Robustheit relativ gut. Wenn beispielsweise kein Kunde in der Bar gefunden wird, dann weiss der Roboter, was er nun machen muss. Dasselbe wenn jemand kein Getränk bestellen möchte. Auch innerhalb der Gespräche ist eine gewisse Flexibilität vorhanden. Jedoch kann hier der Agent auch zum Scheitern gebracht werden, wenn jemand sich nicht auf ein Gespräch einlässt oder nur nicht kontext-relevante Aussagen macht. Ebenfalls kann eine nicht angenommene Veränderung der Umgebung den Agenten zum Scheitern bringen. Für die Auswahl der Aktionen wurde auf eine Nutzensfunktion verzichtet.

4.4 Risiken und Unsicherheiten des programmierten Verhaltens

Da ein externer Service, der in einer Cloud läuft, verwendet wird, muss eine Internetverbindung vorhanden sein. Nebst diesem Risiko stellt auch die vorhandene Software des Pepper ein Risiko dar, da manche Funktionsaufrufe der Schnittstelle nicht immer zur genau gleichen Bewegung beziehungsweise Aktion geführt haben. Zudem konnten nicht alle möglichen Fälle, die zu einer Exception führen könnten, entsprechend behandelt werden, was ebenfalls eine Unsicherheit darstellt. Ein weiteres Risiko birgt die Spracherkennung des Roboters. Obschon mehrere Vorkehrungen getroffen wurden, kann es vorkommen, dass mehr als eine Subscription auf "WordRecognized" zur selben Zeit aktiv sind, was zu Problemen beim Erkennen der Spracheingabe führen könnte.

4.5 Verbesserungspotenzial

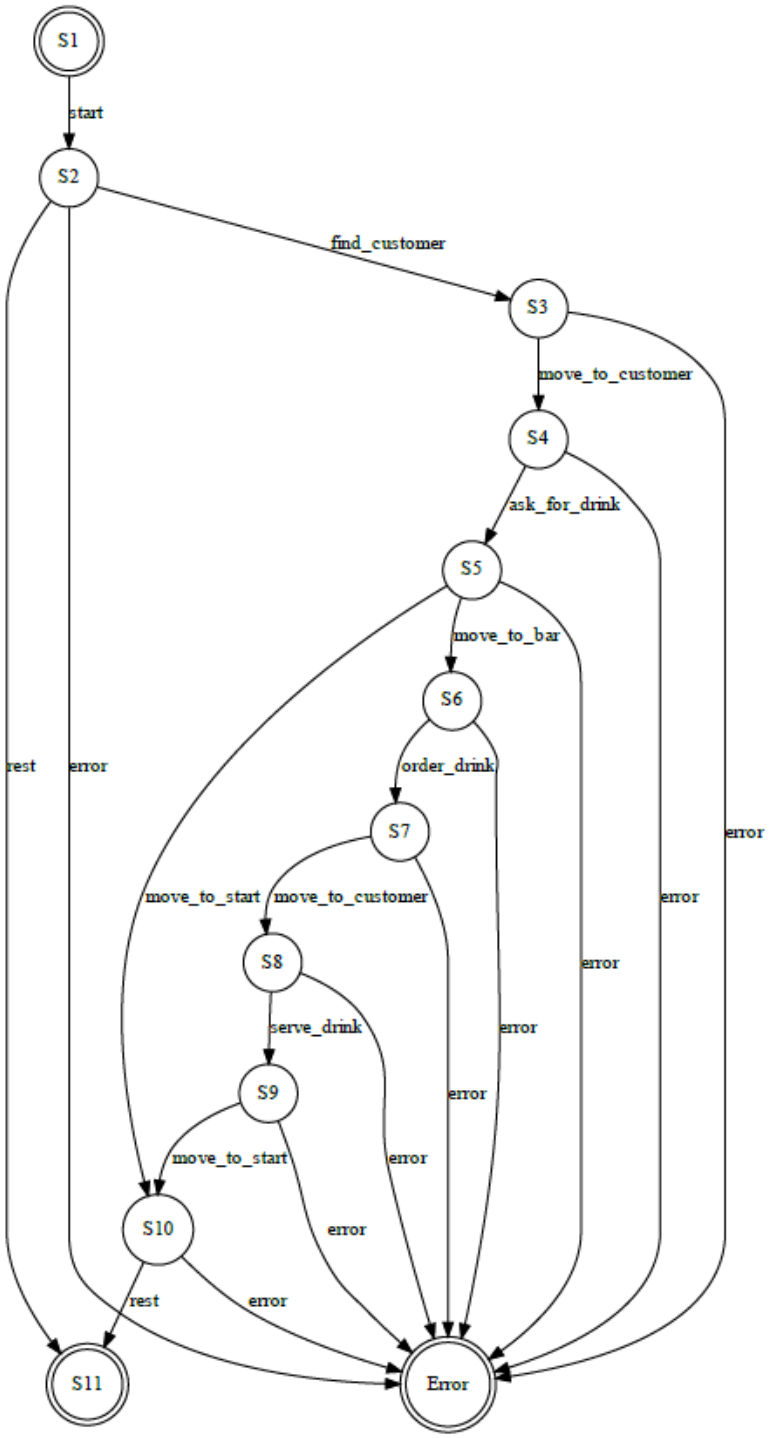
Wenn das Ziel der Einsatz in einer echten Bar ist, muss unter anderem das Wiederfinden des Kunden, wenn sich dieser in der Zwischenzeit bewegt hat, implementiert werden. Dazu gehört dann auch, dass der Roboter Gesichter lernt und sich diese merken kann. Zudem wäre auch die Erkennung, ob ein Kunde bereits ein Getränk hat, eine notwendige Verbesserung. Bei der Interaktion mit Personen wären auch Verbesserungen denkbar, etwa mit dem Einsatz einer externen NLP-Engine. Schlussendlich müsste auch noch die Robustheit weiter verbessert werden, um in einer echten Bar zu bestehen.

5 Reflexion

Bisher mussten in anderen Modulen oft nur einzelne Sensoren angesprochen werden und daher war die Erfahrung mit einem Roboter, der eine Vielzahl von Sensoren und Aktuatoren hat, eine neue und spannende Erfahrung. Selbstverständlich hätten die Autoren gerne den einen oder anderen Punkt aus dem Verbesserungspotenzial bereits umgesetzt, was aber aus Zeitgründen nicht möglich war. Schlussendlich sind die Autoren sehr zufrieden mit «Pepper the waiter» und werden von den gemachten Erfahrungen auch in Zukunft profitieren können.

6 Anhang

6.1 Finite State Machine



Legende States

S1 = Initial
S2 = Waiting_for_input
S3 = Finding_customer
S4 = Moving_to_customer
S5 = Asking_customer_for_a_drink
S6 = Moving_back_to_the_bar
S7 = Ordering_drink_at_the_bar
S8 = Moving_back_to_the_customer
S9 = Serving_drink_to_customer
S10 = Moving_back_to_starting_point
S11 = Resting