ESP/AssB WS15

Aus Progwiki
< ESP

Inhaltsverzeichnis

Einleitung

Das Ausbesserungsbeispiel (AssB) *ersetzt* die Punkte des Hauptbeispiels (AssA). Es sind jene TeilnehmerInnen zugelassen, die auf AssA *weniger als die Hälfte der möglichen Punkte* (<10) erreicht haben. Weiters muss die AssA-Abgabe als *gültiger Versuch* gewertet worden sein.

Bei dieser Aufgabe soll ein Simulator/Debugger für Turingmaschinen erstellt werden. Die Turingmaschine (Band, Position des Schreiblesekopfes, Anfangszustand, Regeln) wird durch eine Textdatei beschrieben und die Simulation soll schrittweise erfolgen können. Weiters sollen verschiedene Arten von Haltepunkten unterstützt werden um zu speziellen Zeitpunkten während der Simulation beispielsweise das Band der Turingmaschine anzeigen zu können.

Programmaufruf

Es gibt genau eine gültige Möglichkeit zum Starten des Programmes. Das Programm erhält als Parameter den Dateinamen der Datei, welche die Turingmaschine beschreibt:

Bsp:

```
./assb tm.txt
```

Dateiformat

Das Dateiformat hat folgenden Aufbau:

```
1. Zeile: Band
2. Zeile: Position des Schreiblesekopfes (Ganzzahl)
3. Zeile: Anfangszustand (Ganzzahl)
Ab Zeile 4: Die Regeln (Übertragungsfunktion) der Turingmaschine im Format:

```

```
<Aktueller Zustand (Ganzzahl)> <Gelesenes Symbol (1 Zeichen)> <Zu schreibendes Symbol (1 Zeichen)>
<Folgezustand (Ganzzahl)> <Bewegung des Schreiblesekopfes (L=Bewegung nach links, R = Bewegung nach Rechts, 0 = Verbleiben)>
```

Beispieldatei:

```
11000
0
1
1 1 0 2 R
1 0 0 6 0
2 1 1 2 R
2 0 0 3 R
3 1 1 3 R
3 0 1 4 L
4 1 1 4 L
4 0 0 5 L
5 1 1 5 L
5 0 1 1 R
```

Die Position des Schreiblesekopfes bezieht sich auf die erste Zeile (Band). Im obigen Beispiel ist die Position 0 und der Schreiblesekopf zeigt daher auf den ersten 1er von "11000". Es soll angenommen werden, dass links und rechts der durch die Datei definierten Symbole unendlich viele Leersymbole stehen. Das Leersymbol wird in der Datei durch Underscore "_" dargestellt. Wäre die Position bei obigen Beispiel kleiner 0 oder größer 4 würde sich der Schreiblesekopf auf einem Leersymbol befinden.

Eine Regel "1 _ 0 2 R" bedeutet: Falls im Zustand 1 ein Leersymbol (_) gelesen wird, dann schreibe an die aktuelle Position des Schreiblesekopfes eine 0, wechsele in den Zustand 2 und bewege den Schreiblesekopf nach rechts.

Hinweis: Es kann angenommen werden, dass die einzelnen Teile einer Regel durch genau ein Blank getrennt sind sodass einfach zB mit fscanf eingelesen werden kann.

Befehle

Nach dem erfolgreichen Aufruf des Programmes soll ein Eingabeprompt "esp> " erscheinen. Zuvor muss allerdings

noch geprüft werden ob die Datei in Ordnung ist. (siehe Rückgabewerte u. Fehlermeldungen weiter unten)

Danach wartet das Programm auf die Eingabe von Befehlen. Ungültige Befehle sollen ignoriert werden. Alle Befehle sind case sensitiv zu interpretieren.

list

Durch Eingabe von "list" sollen die Regeln der Turingmaschine wie folgt ausgegeben werden:

```
esp> list
>>> 1 1 -> 0 2 R
1 0 -> 0 6 0
2 1 -> 1 2 R
2 0 -> 0 3 R
3 1 -> 1 3 R
3 0 -> 1 4 L
4 1 -> 1 4 L
4 0 -> 0 5 L
5 1 -> 1 5 L
5 0 -> 1 1 R
esp>
```

Die Reihenfolge von oben nach unten wie links nach rechts entspricht der Datei. Man beachte, dass die Regel die als nächstes ausgeführt werden soll durch ">>> " am Anfang gekennzeichnet werden muss. Weiters soll zur schöneren Darstellung ein "->" nach dem gelesenen Symbol der Regel eingefügt werden.

step

Der Befehl "step" soll die nächste Regel ausführen und diese ausgeben.

```
esp> step
1 1 -> 0 2 R
esp>
```

show

Durch "show" wird das Band der Turingmaschine ausgegeben. Der Schreiblesekopf wird durch "> <" dargestellt. Nachdem das Band der Turingmaschine theoretisch unendlich lang ist soll das Band nur vom am weitest linken nicht leeren Symbol bis zum am weitest rechten nicht leeren Symbol ausgegeben werden. Befindet sich der Schreiblesekopf auf einem Leersymbol weiter links vom ersten bedruckten Zeichen soll von der Position des Schreiblesekopfes ausgegeben werden. Gleiches gilt für rechts.

```
esp> show
>1<|1|0|0|0
esp>
```

Falls sich der Schreiblesekopf auf Position -3 befindet:

```
>_<|_|_|1|1|0|0|0
```

Falls sich der Schreiblesekopf auf Position 6 befindet:

```
1|1|0|0|0|_|>_<
```

continue

Mittels "continue" soll das Programm bis zum nächsten Haltepunkt oder Programmende ausgeführt werden.

break

Es sollen unterschiedliche Arten von Haltepunkten unterstützt werden:

- break state <zustand>

Es soll angehalten werden wenn sich Turingmaschine im angegebenen Zustand befindet.

Bsp (gilt für die oben beschriebene Datei):

```
esp> break state 2
esp> continue
esp> show
0|>1<|0|0|0
esp>
```

- break write <Symbol>

Es soll angehalten werden wenn (bevor) das angegebene Symbol geschrieben werden soll. Das Beispiel zeigt, dass der Haltepunkt erst ab der nächsten Regel greifen soll und nicht mehr bei der aktuellen (erstes list). Da hier ja schon sowieso gehalten wurde.

```
esp> list
>>> 1 1 -> 0 2 R
 1 0 -> 0 6 0
 2 1 -> 1 2 R
 2 0 -> 0 3 R
 3 1 -> 1 3 R
 3 0 -> 1 4 L
 4 1 -> 1 4 L
 4 0 -> 0 5 L
 5 1 -> 1 5 L
 5 0 -> 1 1 R
esp> break write 0
esp> continue
esp> list
 1 1 -> 0 2 R
 1 0 -> 0 6 0
 2 1 -> 1 2 R
>>> 2 0 -> 0 3 R
 3 1 -> 1 3 R
 3 0 -> 1 4 L
 4 1 -> 1 4 L
 4 0 -> 0 5 L
 5 1 -> 1 5 L
```

```
5 0 -> 1 1 R
esp> show
0|1|>0<|0|0
esp>
```

- break read <Symbol>

Es soll angehalten werden wenn das angegebenen Symbol gelesen wird.

Bsp:

```
esp> break read 0
esp> continue
esp> show
0|1|>0<|0|0
esp>
```

- break pos <Position>

Es soll angehalten werden wenn sich der Schreiblesekopf an der angegebenen Position befindet.

Bsp:

```
esp> break pos 4
esp> continue
esp> show
1|0|0|1|>0<
esp>
```

Alle Haltepunkte sind als temporär zu verstehen. Das bedeutet falls die Bedingung für einen Haltepunkt zutrifft soll dieser Haltepunkt wieder gelöscht werden. Treffen mehrere zu, sollen sämtliche Haltepunkt für die die Bedingung erfüllt war gelöscht werden. Die Haltepunkte sollen bei zutreffender Bedingung sowohl bei continue als auch bei step

gelöscht werden.

quit

Durch Eingabe von "quit" soll das Programm beendet werden. Selbiges gilt bei EOF (Eingabe von ctrl+D am pluto, auch bei EOF soll noch "Bye.\n" ausgegeben werden)

Vor Programmende soll hier noch "Bye.\n" ausgegeben werden.

```
esp> quit
Bye.
```

Programmende

Falls keine weitere Regel mehr angewendet werden kann soll das Programm beendet werden. Zuvor soll noch "machine stopped in state %d\n" ausgegeben werden, wobei für %d der Zustand eingesetzt werden soll, den die Turingmaschine am Ende erreicht hat. Weiters soll noch einmal das Band gleich wie bei "show" ausgegeben werden. Haltepunkte auf den Endzustand sollen nicht mehr greifen, da hier das Programm ja schon beendet werden soll und keine Befehle mehr Sinn machen würden.

Bsp:

```
esp> continue
machine stopped in state 6
1|1|>0<|1|1
```

Rückgabewerte und Fehlermeldungen

Falls das Programm korrekt endet (durch "quit" bzw eof oder keine Regel ist mehr anwendbar):

Rückgabewert: 0

Falls das Programm nicht mit genau einem Parameter gestartet wurde:

```
[ERR] usage: ./assb <file>\n
```

Rückgabewert: 1

Sollte während der Laufzeit der Speicher ausgehen:

```
[ERR] out of memory\n
```

Rückgabewert: 2

Falls die Datei ungültigen Inhalt hat (zB fehlender Inhalt, Position o. Zustand sind keine Zahlen, ...):

```
[ERR] parsing of input failed\n
```

Rückgabewert: 3

Falls die Datei nicht gelesen werden kann (Datei kann nicht geöffnet werden):

```
[ERR] reading the file failed\n
```

Rückgabewert: 4

Falls die Datei zwar syntaktisch richtig ist, es sich jedoch um eine nicht deterministische Turing Maschine handelt:
(Falls für Zustand und gelesenes Symbol mehr als eine Regel zutreffen würde)

```
[ERR] non-deterministic turing machine\n
```

Rückgabewert: 5

Spezifikation

Bitte die folgenden Punkte unbedingt einhalten, da diese einerseits das Testen erleichtern und weiters für die Lernziele notwendig sind.

Einschränkungen

- keine weiteren Ausgaben
- alle Ausgaben müssen auf stdout erfolgen
- Für die Verwaltung des Bandes der Turingmaschine muss dynamischer Speicher verwendet werden. Das Band muss beliebig vergrößert werden können.
- Mindestens eine eigene struct muss sinnvoll eingesetzt werden.

Erlaubte Bibliotheken

- alle C Standard Bibliotheken

Abgabe

- Dateinamen laut Abgabenliste
- Archiv beinhaltet keine Verzeichnisse oder andere Dateien

Abgabenliste

Bis Freitag 19.02.2016 19:00:00:

- Quellcode (assb.c)

in einem .tar.gz oder .zip Archiv (assb.tar.gz oder assb.zip).

Verantwortliche Tutoren

- Christoph Maurer