

Weitere Fragen zu den Übungen

Mit der Bearbeitung der Aufgaben auf dem Übungsblatt sind noch lange nicht alle Fragen gestellt bzw. geklärt. Bei den hier gestellten Fragen geht es nicht um eine korrekte oder inkorrekte Erstellung eines LLWin-Programms, sondern sie sollen als Anregung zu einer tieferen Beschäftigung mit den Aufgaben dienen. Ähnliche Fragen wird man sich bei der Entwicklung eigener Projekte auch stellen müssen, um zum Beispiel die korrekte Abarbeitung der Programme zu gewährleisten (Stichwort: *Programmverifikation*).

(1) Verkehrsampel

- Eine Baustellenampel besteht meistens aus zwei einzelnen Einheiten. Wie müssen sich welche Phasen der beiden Ampeln überlappen? Wie viele Zustände gibt es dann? Wie wirkt sich z.B. die Länge der Baustelle auf die Zustandsübergänge aus?
- Die Wahl der Zykluszeiten ist recht willkürlich. Wie lange sind die Zeiten „in der Natur“ wirklich?
- Will man den Programmablauf als Text festhalten, wie lautet dieser dann?
- Wie lässt sich das Programm unnötig aufblähen, bzw. wie lassen sich Befehle einsparen? (Stichworte: *Ausführungszeit, Speicherverbrauch*)

(2) Bedarfsgesteuerte Fußgängerampel

- Wie viele Zustände hat der Ampel-Zyklus?
- Was passiert, wenn der Fußgänger innerhalb der Auto-Mindest-Grünzeit auf den Taster drückt?
- Stell dir vor, beide Teilampeln wären unabhängige Objekte (zum Beispiel zwei unabhängige Programme oder Programmfäden), die untereinander Botschaften austauschen müssten. Welche Nachrichten wären zu welcher Zeit notwendig? Welche Informationen sind irrelevant für das jeweils andere Objekt?
- Wie sähe die Ampelsteuerung aus, wenn sie zum Beispiel in einer Fußgängerzone platziert ist und nur dann für Autos auf grün springt, wenn wirklich ein Auto kommt, ansonsten aber die Fußgänger passieren lässt?

(3) Garagentor

- Gibt es einen undefinierten Zustand, das heißt ein Zustand, für den in der Aufgabenstellung keine Handlung vorgesehen ist? Wie kann das Programm auf einen solchen Zustand reagieren? Was könnte man als optimistische, was als pessimistische Lösung bezeichnen? Welche Lösung ist aufgrund des Kontextes vorzuziehen? (Stichwort: *undefinierter Zustand, s.u.*)
- Was passiert, wenn E1 gedrückt wird, bevor das Garagentor ganz geöffnet bzw. geschlossen wurde?
- An M2 wird eine gelbe Lampe angeschlossen, die blinken soll, sobald das Tor sich öffnet bzw. schließt. Wie lässt sich eine solche Lampen-Steuerung am geschicktesten nachträglich dem Programm hinzufügen? Ist es möglich diese Steuerung so zu programmieren, dass keine Eingriffe in den ursprünglichen Steuerungsfaden notwendig sind? (Stichwort: *Quellcode-Wartung*)

- Wie ließe sich das Modell und die Steuerung erweitern, dass z.B. beim Schließen des Tores der Motor gestoppt wird, sobald ein Objekt das Schließen verhindert?

(4) Transportband

- Was passiert, wenn ein Gepäckstück versehentlich vom Transportband fällt? (Stichwort: *Timeout*)
- Was geschieht, wenn kurz hintereinander auf beiden Seiten Gepäckstücke auf das Förderband gelegt werden?
- Wie sieht die Situation aus, wenn dies gleichzeitig geschieht?
- Was ereignet sich, wenn mehrere Gepäckstücke auf das Förderband gelegt werden?
- Welche anderen Sensoren würden sich eignen, um die Lichtschranken zu ersetzen? Sind diese stärker oder weniger stark fehleranfällig? Welche Vor- oder Nachteile haben andere Sensoren?

Zum Stichwort „Undefinierter Zustand“

Während man landläufig davon ausgeht, dass z.B. eine Aussage entweder „Wahr“ (True; 1; L; ...) oder „Falsch“ (False; 0; ...) ist, gibt es in vielen Fällen nicht die Möglichkeit, zwischen diesen beiden Zuständen zu unterscheiden, das Ergebnis der Aussage ist also vielmehr undefiniert.

Dieser Zustand wird oftmals mit dem Zeichen \perp (genannt „Bottom“ *engl. für „Grund, Boden, Unterseite“*) bezeichnet.

Beispiele für solche undefinierte Aussagen:

- Ist es in der Dämmerung „hell“ oder „dunkel“?
- Ist ein Glas, das zur Hälfte gefüllt wurde, als „voll“ oder als „leer“ zu bezeichnen?
- Welches Ergebnis hat der Term „ $267 / (24 - 2 * 12)$ “ ?

Bei der Verknüpfung mehrerer Aussagen, zum Beispiel mit „UND“ („AND“, \wedge) bzw. „ODER“ („OR“, \vee) kann mit einer undefinierten Teilaussagen unterschiedlich verfahren werden, denn teilweise genügt schon die Auswertung eines Teils, um das gesamte Ergebnis zu bestimmen („1 ODER x“ ist immer „1“, egal was „x“ ist). Bricht man nun in einem solchen Fall mit der Auswertung ab (schnellere Programmausführung), so nennt man diese Auswertungsmethode „nicht strikt“, gibt man als Ergebnis \perp zurück, sobald ein Teilterm \perp ist, so nennt man die Auswertung „strikt“.

Beispiele:

	$0 \wedge \perp$	$1 \wedge \perp$	$\perp \wedge 0$	$\perp \wedge 1$	$0 \vee \perp$	$1 \vee \perp$	$\perp \vee 0$	$\perp \vee 1$
strikt	\perp	\perp	\perp	\perp	\perp	\perp	\perp	\perp
nicht strikt	0	\perp	\perp	\perp	\perp	1	\perp	\perp