Universität Bielefeld

Technische Fakultät
Arbeitsgruppe Rechnernetze und Verteilte Systeme

Diplomarbeit im Studienfach

Naturwissenschaftliche Informatik

Spezifikation eines 20 l-Perfusionsbioreaktor in TLA$^+$

vorgelegt von
Lutz Sommerfeld

11. Juli 1997

Erstgutachter: Prof. Dr. Peter Ladkin
Arbeitsgruppe Rechnernetze und Verteilte Systeme
Technische Fakultät
Universität Bielefeld

Zweitgutachter: Dr. Dirk Lütkemeyer
Arbeitsgruppe Zellkulturtechnik
Technische Fakultät
Universität Bielefeld

Inhaltsverzeichnis

Vorwort

1 Einleitung und Problemstellung **1**
1.1 Einleitung 1
1.2 Problemstellung 2

2 Der 20 Liter Perfusionsbioreaktor **4**
2.1 Zellkultur 4
2.2 Kulturmedium 4
2.3 Bioreaktor 5
2.3.1 Sterilisation 6
2.3.2 Befüllung und Entnahmen 6
2.3.3 Rührer und Begasung 7
2.4 Regelung der Begasungsanlage 8

3 Einführung in TLA und TLA$^+$ **10**

4 Spezifikation des Bioreaktors **13**
4.1 Interviews 13
4.2 Modularisierung 14
4.3 Zustandsvariablen und Konstanten 15
4.4 Modul MiscDef 15
4.5 Modul Echtzeit 16
4.6 Modul Umgebung 18
4.7 Modul Medium 18
4.8 Modul Bioreaktor 21
4.9 Modul Kontrolle 25
4.10 Modul Kultivierung 28

5 Spezifikation der Regelungssoftware **32**
5.1 Softwareregelung vs. Analogen PID-Regler 33

5.2 Zustandsvariablen und Konstanten 33
5.3 Modul Sensorprozess 34
5.4 Modul Aktorprozess 35
5.5 Modul Regelung 36
5.6 Modul SoftwareKontrolle 39

6 Beurteilung der Regelungssoftware 40
6.1 Anwendbarkeit der Spezifikationen 40
6.2 Refinement Mapping 41
6.3 Sicherheitseigenschaften 42
6.4 Ergebnis 45

7 Zusammenfassung und Ausblick 46

Anhang

A Die Module zum Bioreaktor 48
A.1 Modul MiscDef 48
A.2 Modul Echtzeit 49
A.3 Modul Umgebung 50
A.4 Modul Medium 51
A.5 Modul Bioreaktor 53
A.6 Modul Kontrolle 57
A.7 Modul Kultivierung 61

B Die Module zur Regelungssoftware 62
B.1 Modul Sensorprozess 62
B.2 Modul Aktorprozess 64
B.3 Modul Regelung 65
B.4 Modul SoftwareKontrolle 67

C Tabellen 68
Zustandsvariablen Software 68
Konstanten Software 69
Zustandsvariablen Bioreaktor und Kultivierung 70
Konstanten Bioreaktor und Kultivierung I 71
Konstanten Bioreaktor und Kultivierung II 72

Literaturverzeichnis 73

Erklärung 75

Vorwort

Die interdisziplinäre Ausrichtung des Studiengangs Naturwissenschaftliche Informatik spiegelt sich in dieser Diplomarbeit wider. Auf Initiative der beiden Arbeitsgruppen Zellkulturtechnik sowie Rechnernetze und Verteilte Systeme der Technischen Fakultät ist diese Arbeit entstanden. In der Hoffnung, mit dieser Arbeit beiden Arbeitsgruppen bei ihrer weiteren Forschungsarbeit behilflich zu sein, möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Diplomarbeit durch ihre Anregungen und Hinweise unterstützt haben. Insbesondere gilt mein Dank Peter Ladkin für die intensive Betreuung während der gesamten Entwicklungszeit sowie für die Unterstützung bei der auszugsweisen Präsentation dieser Arbeit auf dem 7. GI/ITG-Fachgespräch „Formale Beschreibungstechniken für verteilte Systeme“ im Juni 1997 in Berlin. Einen herzlichen Dank möchte ich auch Dirk Lütkemeyer aussprechen für seine ständige Bereitschaft, mir meine Fragen zum Bioreaktor zu beantworten und mich mit der entsprechenden Fachliteratur zu versorgen.

Bielefeld, den 11. Juli 1997 Lutz Sommerfeld

Kapitel 1

Einleitung und Problemstellung

1.1 Einleitung

Mikroprozessoren sind aus dem modernen Alltag nicht mehr wegzudenken. Sie sind in fast allen Bereichen des Lebens zu finden. Viele technische Produkte, die wir täglich gebrauchen, funktionieren nur mittels Computerunterstützung. Spielzeuge, Autos, Flugzeuge, Rettungsdienste, Forschungseinrichtungen, fast alle Einrichtungen der Telekommunikation ...basieren auf dem zuverlässigen Funktionieren ihrer digitalen Komponenten und deren fehlerfreien Programmierung.

Das fehlerfreie Funktionieren der Computereinheit ist je nach Anwendungsgebiet von enormer Wichtigkeit. Ein fehlerhaftes Spielzeug kann normalerweise keinen größeren Schaden anrichten, jedoch muß ein Passagierflugzeug, welches nur noch mittels einer Computersteuerung geflogen werden kann, mit einem fehlerfreien Datenverarbeitungssystem ausgestattet sein. Es hat sich in den letzten Jahren gezeigt, daß insbesonder das fehlerfreie Funktionieren der Software noch nicht ausreichend gewährleistet werden kann[1]. Diese Softwareunsicherheit ist größtenteils begründet, in der in der Industrie bevorzugten Fehlererkennung durch Testen. Mit dem Testen von Software kann allerdings nur die Anwesenheit von Fehlern, nicht aber deren Abwesenheit gezeigt werden. Daß eine Software einen Test fehlerfrei überstanden hat, kann nicht als ein Zeichen von tatsächlicher Fehlerfreiheit verstanden werden. Solange nicht *alle* möglichen Fälle getestet worden sind, ist es immer möglich, daß ein vorhandener Fehler bisher noch nicht gefunden ist. Daher ist es wichtig, neue Techniken der Fehlerbeseitigung zu entwickeln.

[1] Bei der Untersuchung von Flugzeugunglücken der letzten Jahre in [Lad97.2] wurde z.T. fehlerhafte Software als Absturzursache ermittelt.

Studien, z.B. [Lev95], zu Folge basiert ein sehr großer Teil aller Softwarefehler auf einer fehlerhaften Spezifikation. Daher ist in den letzten Jahren eine große Anzahl von formalen Spezifikationstechniken (FDT[2]) entwickelt worden. Durch eine Formalisierung der Spezifikationen soll verhindert werden, daß schon in der Spezifikation Fehler entstehen können, welche sich in die entgültige Software fortpflanzen. Wichtig für diese neuen FDT's ist es, daß sie Mechanismen zur Fehlerbeseitgung aufweisen, die auf die Spezifikationen angewendet werden können. Nur FDT's, die diese Möglichkeiten aufweisen, sind wirklich zur Fehlerbeseitigung bei der Implementation von Problemen und damit zur Steigerung der Softwaresicherheit geeignet. Die in dieser Arbeit verwendeten FDT, TLA$^+$, ermöglicht es, einen formalen logischen Beweis zu führen, daß von zwei Spezifikationen eine die andere implementiert. Im Falle von TLA$^+$ entspricht Implementation Implikation. In [Lad95] wird ein solcher Beweis geführt.

Damit bietet TLA$^+$ die geforderten Mechanismen, die zur Erkennung und Beseitigung von Implementationsfehlern notwendig sind, und gewährleistet auf diese Weise Fehlersicherheit in der Spezifikation.

1.2 Problemstellung

Um eine möglichst universelle Nutzung einer FDT zu gewährleisten, sollte der Aufbau der Sprache nicht auf bestimmte Nutzungsgebiete beschränkt sein. Daher sollte auch TLA$^+$ die Möglichkeit bieten, Beschreibungen von Problemen aus völlig unterschiedlichen Anwendungsgebieten zu ermöglichen. Es hat sich gezeigt, daß TLA$^+$ sowohl in der Beschreibung von Problemen der Informatik selbst [Lad96.1, Lazy Cache Problem], als auch bei hybriden Systemen [Lam93, Gasburner] einsetzbar ist. Die in diesem Zusammenhang erforderliche Behandlung von Echtzeit stellt für TLA$^+$ kein Problem dar. In [AL92] wird gezeigt, wie Echtzeitbedingungen in TLA$^+$ behandelt werden können. Die Anwendung von TLA$^+$ in der Beschreibung von Problemen in der Nachrichtentechnik [MFJ97, RLP Kommunikationsprotokoll] und [Hen97] als auch in der Regelung von physikalischen Geräten [ABL96, Steamboiler] hat gezeigt, daß TLA$^+$ in unterschiedlichsten Gebieten eingesetzt werden kann.

Alle bisherigen Anwendungen weisen jedoch als Gemeinsamkeit auf, daß der formalen Beschreibung eine *metaformale* Problembeschreibung zugrunde

[2]FDT: Formal Description Technique

liegt. *Metaformal* heißt, daß alle problemrelevanten Details, insbesondere die Einflüsse der Umwelt als auch der einzelnen Komponenten aufeinander, lückenlos bekannt und beschrieben sind. Meist kann jedoch nicht davon ausgegangen werden, daß zu einem realen Problem alle Zusammenhänge vollständig bekannt sind und in einer abschliessenden Weise benannt werden können. Bei dem in dieser Arbeit behandelten Perfusionsbioreaktor und der in ihm stattfindenden Zellkultivierung handelt es sich um einen Prozeß, bei dem nicht alle Zusammenhänge vollständig bekannt sind.

Um Einsichten in die Nutzbarkeit von TLA$^+$ für solche *vage* beschriebenen Problemstellungen zu erhalten, wird aufbauend auf den Informationen aus mehreren Experteninterviews eine TLA$^+$ Spezifikation des Bioreaktors erstellt. In Verbindung mit der TLA$^+$ Beschreibung einer existierenden Regelungssoftware soll anschließend ermittelt werden, welche Schlüsse über die Softwareimplementation der Begasungsregelung sich aus der Reaktorbeschreibung ergeben und ob TLA$^+$ geeignet ist, Systeme, zu denen nur eine vage Beschreibung vorliegt, zu spezifizieren.

Kapitel 2

Der 20 Liter Perfusionsbioreaktor

Der 20 Liter Perfusionsbioreaktor, nachfolgend nur Reaktor genannt, ist auf der Abbildung 2.1 auf der nächsten Seite zu sehen. Er wurde 1986 u.a. von Prof. Jürgen Lehmann, AG Zellkulturtechnik, speziell zur Kultivierung tierischer Zellen entwickelt.

2.1 Zellkultur

Der Reaktor eignet sich zur Kultivierung von Tierzellen verschiedenster Arten, wie sie in Forschung und Industrie verwendet werden. In den meisten Fällen handelt es sich um immortalisierte Zellen, die nicht altern und daher ohne äußere schädliche Einflüsse nicht absterben. Dies gewährleistet über einen langen Zeitraum eine extrem geringe Anzahl an toten Zellen im Kulturmedium. Auch pflanzliche Zellen lassen sich mit dem Reaktor kultivieren. Der Nutzen der Kultivierung besteht zumeist nicht in der Vermehrung der Zellen an sich, sondern in dem Nebenprodukt, welches von den Zellen während des Wachstums produziert und nach der Zellernte aus dem Kulturüberstand extrahiert wird.

2.2 Kulturmedium

Ohne ihre natürliche Umgebung können Zellen nicht überleben. Um diese nachzubilden, werden die Zellen in ein Basalmedium gegeben, welches die für die jeweilige Zellinie überlebensnotwendigen Nährstoffe enthält. Zusätzlich sorgt es bei einer entsprechenden Begasung für den gewünschten Umgebungs-pH-Wert. Der osmotische Druck des Basalmediums entspricht dem der Zelle, um ein Platzen zu verhindern.

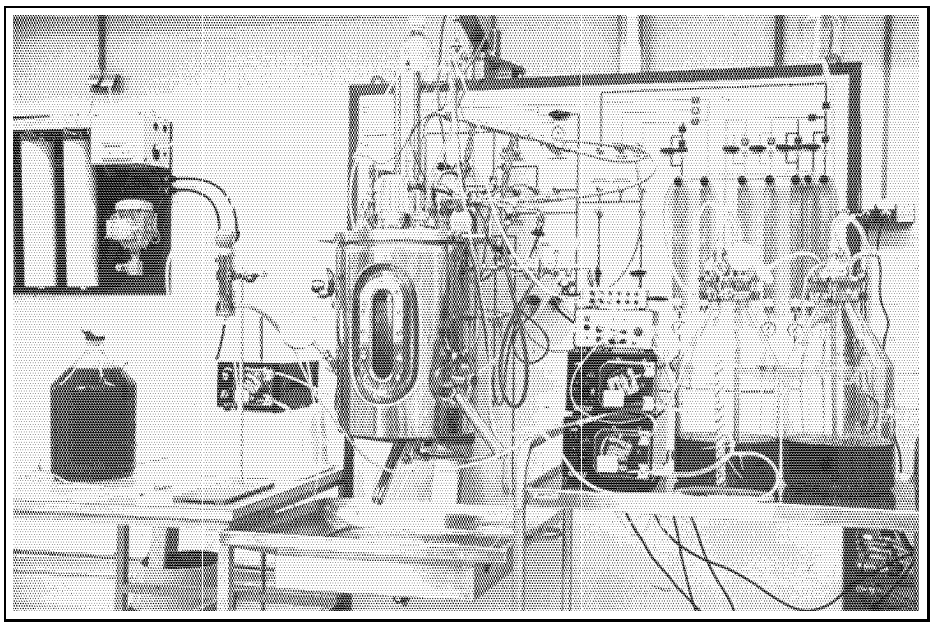

Abbildung 2.1: 20 Liter Perfusionsbioreaktor in Betrieb. Dargestellt sind der Kessel, verschiedene Vorratsgefäße für Medium und Pumpen

Während des Wachstums der Zellen werden die Inhaltsstoffe des Mediums verbraucht. Im Austausch gelangen Ausscheidungen der Zellen in das Kulturmedium. Um eine Erneuerung des Basalmediums zu erreichen, ist es bei einem Perfusionsbioreaktor möglich, Medium auszutauschen, während die Zellen im Reaktor verbleiben.

Zur Vorbereitung der Kultivierung im Reaktor muß die Stammkultur in einem kleinen Brutschrankfermenter zu einer solchen Zelldichte herangezüchtet werden, daß sich zu Beginn mindestens $2 \cdot 10^5 \frac{Zellen}{ml}$ im Reaktor befinden.

Die wichtigste Voraussetzung für eine erfolgreiche Kultivierung ist, daß zu keinem Zeitpunkt Bakterien oder Pilze in das Kulturmedium gelangen.

2.3 Bioreaktor

Der Reaktor besteht aus einem ca. 30 Liter fassenden Edelstahlgefäß. Die Arbeitsbefüllung sollte 20 Liter nicht überschreiten. Er ist versehen mit ver-

schiedensten Vorrichtungen zur Begasung und Befüllung. Es existieren Leitungssysteme für Dampf, Warm- und Kühlwasser, welches durch die doppelwandige Außenhülle geleitet werden kann, um den Inhalt zu erwärmen oder abzukühlen.

2.3.1 Sterilisation

Zur Sterilisation des Reaktors wird die doppelwandige Außenhülle mit Wasserdampf gefüllt. Auf diese Art wird Reinstwasser in seinem Inneren aufgeheizt, und es ensteht Wasserdampf von 121° Celsius mit einem Überdruck von ca. 1,2 bar. Nach der Dampfsterilisation (ca. 60 min) wird das Reinstwasser über ein Ablaßventil im Boden des Reaktors entfernt. Dieses Ventil wird anschließend mit Dampf nachsterilisiert.

2.3.2 Befüllung und Entnahmen

Zum sterilen Befüllen des Reaktors gibt es einen Anstechstutzen, der mit einer Kautschuk-Dichtung verschlossen ist. Diese Dichtung wird mit einer Nadel durchstoßen und endet in einem Steigrohr. Mittels dieser Nadel können Stoffe sowohl in den Bioraktor gelangen, als auch aus diesem entnommen werden.

Zur Erstbefüllung wird das Basalmedium über den Anstechstutzen in den Reaktor gepumpt. Um auszuschließen, daß das Basalmedium mit Pilzen oder Bakterien verseucht ist, wird es schon fünf Tage vor der Zugabe der Zellen in den Reaktor gefüllt. Gibt es in dieser Zeit keine Anzeichen einer Verseuchung, so kann mit der Kultivierung begonnen werden, indem die Stammkultur über den Anstechstutzen in das Basalmedium im Reaktor gegeben wird.

Ein Vorteil des Perfusionsbioreaktors ist die Möglichkeit, das Basalmedium im Reaktor zu erneuern. Zu diesem Zweck befindet sich im Inneren eine Membran, die durch ihre Porengröße (ca. $0,2\mu m$) mittels Filtration ein Entnehmen aus und Einfüllen von Basalmedium in den Reaktor ermöglicht, ohne daß Zellen im Reaktor verloren gehen.

Eine Entnahme von sterilen Proben erfolgt über den Anstechstutzen, der auch zur Befüllung des Reaktors dient. Die eingestochene Nadel muß jedoch während der gesamten Kultivierung in der Dichtung verbleiben, und bildet daher ein erhebliches Verseuchungsrisiko. Normalerweise werden keine sterilen Proben benötigt. Daher gibt es zur Entnahme von unsterilen

Proben an der Seite des Reaktors Ventile. Diese lassen sich nach ihrer Benutzung über spezielle Dampfleitungen nachsterilisieren, sodaß fast keine Verseuchungsgefahr bei dieser Art der Entnahme besteht.

Soll die Zellkultur abgeerntet werden, so kann über ein am Boden des Reaktors befindliches Ventil das gesamte Kulturmedium abgelassen werden.

2.3.3 Rührer und Begasung

Ein besonders wichtiger Aspekt der Kultivierung ist die Begasung des Kulturmediums. Sie erfolgt vorrangig über eine Membran, welche um mehrere *Paddel* gewickelt einen Rührer bildet, der zur optimalen Verteilung der Zellen unten mit einer Turbulenzspirale versehen ist. Dieser Rührer sorgt durch seine Taumel-Drehbewegung für eine gleichmäßige Verteilung aller Inhaltsstoffe, ohne daß durch zu große Scherkräfte die Zellen beschädigt werden. Diese Membranbegasung hat einerseits den Vorteil, daß die Grenzfläche des Gases an der Außenseite der Membran mit dem Kulturmedium besonders groß ist, was zu einem besseren Übergang des Gases in die Flüsigkeit führt. Durch die Wahl eines geeigneten Gasdrucks wird andererseits eine blasenfreie Begasung erreicht, welche notwendig ist, damit die Zellen nicht durch aufsteigende Blasen zerstört werden.

Das Gasgemisch setzt sich aus den vier Komponenten Luft, Sauerstoff (O_2), Stickstoff (N_2) und Kohlendioxid (CO_2) zusammen. Mittels CO_2 wird der pH Wert des Kulturmediums reguliert. Luft und O_2 dienen dazu, die Luftsättigung (pO_2) im Medium auf einem bestimmten Sollwert zu halten. Da die Zellen O_2 verbrauchen und in CO_2 umwandeln, ist es mit steigender Anzahl an Zellen notwendig, den CO_2 Anteil zu veringern und den Luft bzw. O_2 Anteil des Gasgemischs zu erhöhen, um einen konstanten pH und pO_2 Wert zu erreichen. Die Zusammenhänge zwischen Zellwachstum und Gasmischung im Kulturmedium sind in der Abbildung 2.2 auf Seite 9 dargestellt. Trotz schwankender Anteile von Luft, O_2 und CO_2 soll dennoch ein konstanter Gasmassendurchfluß gehalten werden. Hierzu bietet sich N_2 als *Füllstoff* an, da es keinen Einfluß auf den Kultivierungsprozeß ausübt. N_2 bewirkt lediglich eine Senkung der Luftsättigung im Medium.

Zusätzlich zur Membranbegasung gibt es noch die Möglichkeit, den Kopfraum oberhalb des Kulturmediums entweder mit Luft oder mit der Membrangasmischung zu begasen.

2.4 Regelung der Begasungsanlage

Die aktuellen Werte von pO_2 und pH im Kulturmedium werden über entsprechende Sonden am Reaktor gemessen und als elektronische Signale an die PID-Reglereinheit geleitet. Diese berechnet aus der entsprechenden Abweichung vom Sollwert die jeweilige Stellgröße. Der Stellwert für pH wird direkt zur Einstellung des CO_2-Anteils in der Membrangasmischung verwendet.

Bei der Umsetzung des pO_2-Stellwertes wird unterschieden zwischen einer Begasung mit Luft/N_2 oder Luft/O_2. Zu Beginn der Kultivierung ist die Begasung mit Luft und Stickstoff ausreichend, um dem Stellwert entsprechen zu können. Mit zunehmender Zellzahl überschreitet jedoch der Stellwert den natürlichen Sauerstoffgehalt der Luft, und die Begasung muß auf den Betrieb mit Luft und Sauerstoff *von Hand* umgestellt werden. Es ist daher erforderlich, den pO_2-Stellwert während der gesamten Luft/N_2 Begasung zu beobachten, um eine Unterversorgung mit Sauerstoff über einen längeren Zeitraum zu verhindern. Das Einhalten eines konstanten Massendurchflusses in der Membran erfordert für pO_2-Stellwerte, die unterhalb der Luftsättigung der Umgebungsluft liegen, daß die Begasungsluft mit N_2 verdünnt wird. Für darüber liegende Stellwerte wird die Aufteilung von Luft und Sauerstoff derart gewählt, daß der Massendurchfluß konstant bleibt.

Im Rahmen einer Diplomarbeit [Bock96] wurde für die Regelung von pH und pO_2 eine Softwaresteuerung entwickelt. Diese ermöglicht eine rechnergestützte und daher flexiblere klassische PID-Regelung[1]. Zusätzlich wurde auch eine pO_2 Fuzzy-Regelung implementiert.

Die Stellwerte für den Druck und die Temperatur im Reaktor werden mittels der PID-Reglereinheit bestimmt und über entsprechende Stellglieder umgesetzt.

[1] Siehe hierzu auch Abschnitt 5.1 auf Seite 33

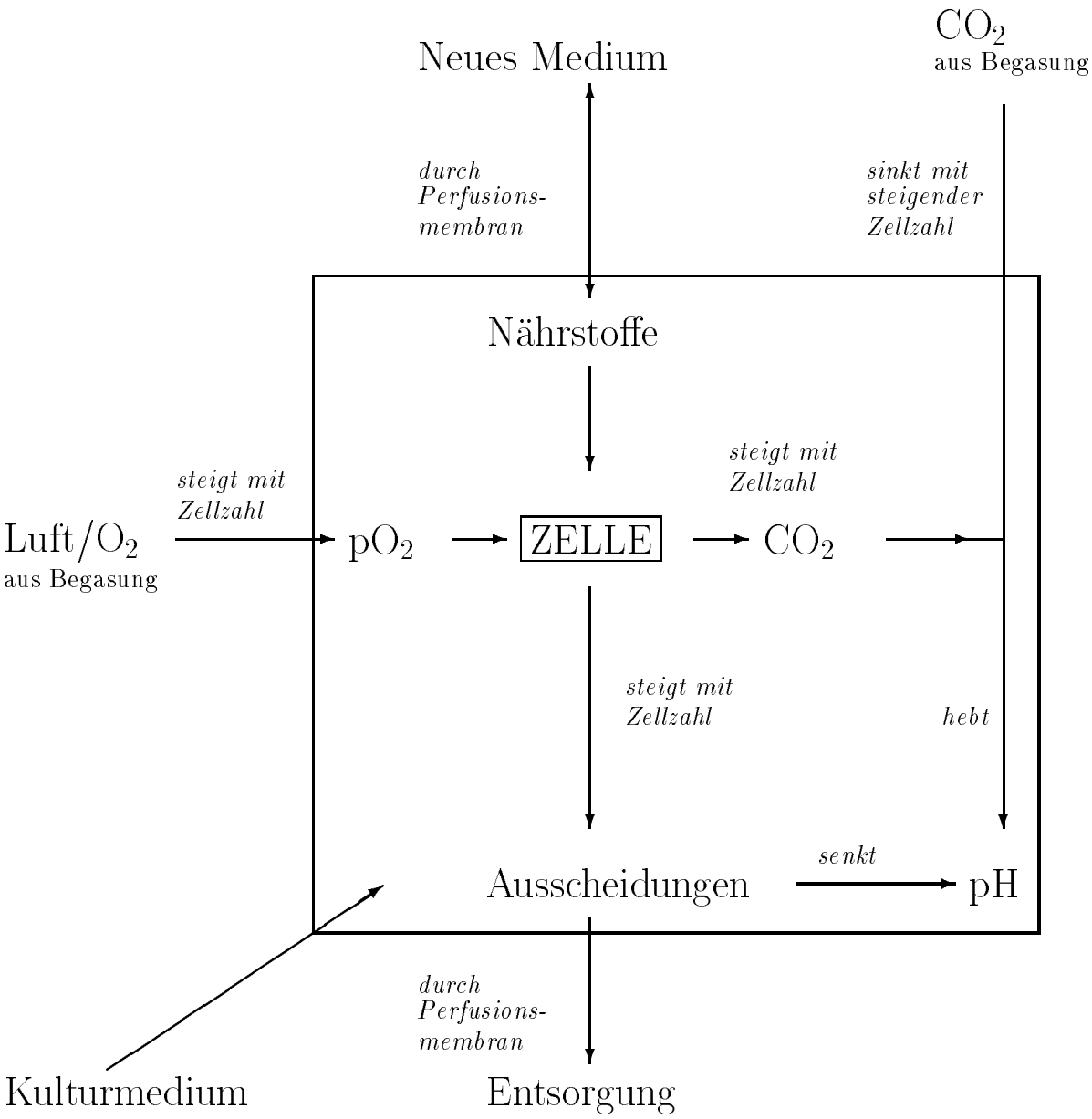

Abbildung 2.2: Zusammenhang zwischen Zellwachstum, pO_2, pH und Begasung

Kapitel 3

Einführung in TLA und TLA$^+$

Bei TLA$^+$ handelt es sich um eine von Lamport [Lam94] entwickelte Spezifikationssprache, die auf der *Temporal Logic of Actions*, TLA basiert. Die Semantik von TLA kann mit dem Modell einer unendlichen Folge diskreter, zeitlich linearer Zustände beschrieben werden. Jeder Zustand definiert und unterscheidet sich über die den Zustandsvariablen zugewiesenen Werte.

TLA Ausdrücke lassen sich in vier Klassen einteilen.

Temporale Ausdrücke enthalten temporale Operatoren, um über eine unendliche Folge von Zuständen Aussagen zu treffen. Die beiden wichtigsten temporalen Operatoren sind $\Box$ und $\exists\!\!\!\exists$. $\Box A$ besagt daß eine Aussage A für alle folgenden Zustände wahr ist. $\exists\!\!\!\exists$ ist der temporale Existenzquantor (nicht zu verwechseln mit dem logischen $\exists$), welcher es erlaubt, Zustandsvariablen innerhalb der Module zu erzeugen[1].

Actions beschreiben den Übergang von einem Zustand in den nächsten.[2] Dem Namen der Variablen, welche den Wert im nächsten Zustand repräsentiert, wird ein *prime* angehängt. So beschreibt z.B. $x' = x + 1$, daß der Wert von x im folgenden Zustand um 1 erhöht ist.
Folgende Bezeichnungen sind zu beachten:

- $[A]_e$ bedeutet, daß entweder A eintritt, oder e ändert seinen Wert nicht.
- $\langle A \rangle_e$ bedeutet, daß A eintritt und e dadurch verändert wird.

[1] Ein *hiding of internal variables* für TLA

[2] Als „nächster Zustand“ wird in TLA der Folgezustand bezeichnet, welcher sich von dem vorhergehenden durch den Inhalt mindestens einer Zustandsvariablen unterscheidet; d.h. TLA ist „stuttering invariant“.

- ENABLED A heißt, A ist möglich, d.h. alle Voraussetzungen zum Erreichen des in A beschriebenen nächsten Zustandes sind erfüllt.
- UNCHANGED e bedeutet, e verändert sich nicht.

Zustands-Ausdrücke treffen Aussagen über einen bestimmten Zustand. Sie enthalten keine *geprimten* Zustandsvariablen, z.B. $y = x + 3$.

Konstante Ausdrücke enthalten nur Konstanten, also keine Zustandsvariablen. [3]

TLA$^+$ erweitert TLA um einige syntaktische Aspekte. Zu diesen gehört unter anderem die Möglichkeit der Modularisierung. Es ist in TLA$^+$ möglich, eine Spezifikationen in mehrere Module zu unterteilen. Diese werden mit den Befehlen **include** und **import** in beliebige andere Module eingebunden. Während **import** einer textuellen Ersetzung gleichkommt, lassen sich mit **include** mehrere Instanzen des selben Moduls unter verschiedenen Namen in einem Modul einbinden. Durch eine entsprechende Substitution des Modulnamens als auch der Bezeichner der Zustandsvariablen aus dem eingebundenen Modul der Form

include *Beispiel* **as** *meinBeispiel* **with** $x \leftarrow meinX$

ist es möglich, auf die eingebundenen Actions und Zustandsvariablen zuzugreifen. In diesem Fall würde eine Action, die im Modul *Beispiel* den Namen *Incr* trägt, durch *meinBeispiel.Incr* angesprochen. Die Zustandsvariable *meinX* aus dem Hauptmodul entspricht in diesem Fall der Zustandsvariablen x aus dem Modul *Beispiel*.

TLA$^+$ basiert auf einer Logik; daher ist eine TLA$^+$ Spezifikation ein logischer Ausdruck, welcher für das beschriebene Problem den Wert TRUE besitzt. Eine TLA$^+$ Spezifikation beschreibt einen Anfangszustand und alle möglichen Veränderungen des Systems von einem Zustand in einen anderen. Auf diese Weise ist die Menge aller möglichen Zustände, die das System erreichen kann, beschrieben. Um eine Implementation auf ihre Korrektheit bezüglich des implementierten Problems zu verifizieren, muß die Implikation

$$implementation \Rightarrow problem$$

[3] Sowohl Konstanten als auch Zustandsvariablen sind im mathematischen Sinne Variablen. Daher können auch Konstanten einen beliebigen Wert annehmen, dieser muß jedoch in *allen* Zuständen unverändert bleiben. Dies unterscheidet sie von den Zustandsvariablen.

bewiesen werden. Mittels eines solchen Beweises wird gezeigt, daß es zu jedem Zustand des Problems einen äquivalenten Zustand in der Implementation gibt. Im Kapitel 6 auf Seite 40 wird näher auf die Einzelheiten eines solchen Beweises eingegangen.

Durch die relativ einfache Semantik ist es mit TLA$^+$ möglich, Problemstellungen aus sehr unterschiedlichen Gebieten formal zu beschreiben. Zusätzlich bietet TLA eine formale Verifikationstechnik für Implementationsbeweise. Die Beweisregeln und eine ausführliche Beschreibung von TLA enthält [Lam94]. Für eine ausführliche Erklärung zu TLA$^+$ sei auf [Lam95] verwiesen.

Kapitel 4

Spezifikation des Bioreaktors

4.1 Interviews

Der Entwurf des Reaktors basiert auf dem Expertenwissen der AG Zellkulturtechnik von Prof. Lehmann. Alle Vorrichtungen und Details des Reaktors haben eine mit der Kultivierung zusammenhängende Bedeutung. Um den Reaktor zu spezifizieren, ist es notwendig, diese Zusammenhänge zu kennen und in die Spezifikation einzubeziehen. Das erforderliche Expertenwissen wurde in mehreren Interviewsitzungen mit Dr. Dirk Lütkemeyer, AG Zellkulturtechnik, zusammengetragen. Dr. Lütkemeyer ist durch seine langjährige Forschungsarbeit mit dem Reaktor bezüglich Aufbau und Nutzung bestens vertraut. Die Aussagen des Dr. Lütkemeyer wurden von mir stichpunktartig aufgezeichnet und dienten anschliessend zur Erstellung der Spezifikation.

Zunächst wurde mir am Bioreaktor der Aufbau und die allgemeine Funktionsweise des Geräts erklärt und am Beispiel der laufenden Kultivierung verdeutlicht. Darauf aufbauend erstellte ich eine erste Zusammenstellung an Zustandsvariablen und folgende Liste von Fragen zu den Zusammenhängen zwischen diesen Zustandsvariablen:

Welche zeitlichen und physikalischen Bedingungen gibt es für

- Abweichungen vom pH Sollwert
- Abweichungen vom pO_2 Sollwert
- Abweichungen vom Temperatur Sollwert
- die Zellzahl und das Zellwachstum
- die Bakterienanzahl und das Bakterienwachstum

- Abweichungen von den Sollwerten der Nährstoffe
- die homogene Verteilung der Zellen
- den Druck im Bioreaktor

Es ergaben sich weitere Interviewsitzungen, in denen detailliert auf Fragen zur Begasung, Befüllung und anderer Einzelheiten eingegangen wurde. Die Zwischenergebnisse der aus den Interviews resultierenden TLA$^+$ Spezifikationen wurden in Gesprächen mit Dr. Lütkemeyer auf die Übereinstimmung mit dem Reaktor und seiner Anwendung überprüft.

Schon während der ersten Interviews stellte sich heraus, daß es zu einigen Fragen keine physikalisch exakten Antworten gibt. Es existieren zwar zahlreiche mathematische Modelle und Formeln für die Zusammenhänge zwischen dem Wachstum der Zellen und z.B. der Sauerstoffversorgung oder der Versorgung mit Nährstoffen[1], jedoch sind diese Formeln mit einer großen Anzahl von systembedingten Parametern versehen, die für den hier vorliegenden Reaktor nicht bestimmt wurden. Das folgende Zitat, welches die Regelung der Gasmischung betrifft, verdeutlicht, daß eine detaillierte oder exakte Kenntnis der Systemparameter für die Durchführung der Kultivierung nebensächlich ist.

> „Solange der pO_2- und pH- Istwert nicht länger als eine Zeit δ vom Sollwert abweicht, verdoppelt sich die Kultur innerhalb von 24 Stunden."

Aussagen solcher Art zusammen mit den physikalischen Merkmalen des Reaktors dienten als Grundlage für die erstellte TLA$^+$ Spezifikation.

4.2 Modularisierung

TLA$^+$ bietet die Möglichkeit, Spezifikationen in Module zu unterteilen. Einerseits ist es auf diese Weise möglich, komplexere Problemstellungen in kleinere Einheiten zu unterteilen, um diese in einer eigenen abgeschlossenen Spezifikation zu beschreiben. Andererseits kann auf diese Weise eine (Teil)Spezifikation beliebig oft an anderer Stelle Verwendung finden, indem sie einfach in die neue Spezifikation eingebunden wird.

Die Unterteilung des Reaktors in mehrere Module erfolgt zunächst gemäß den vorliegenden unterschiedlichen Einheiten, aus denen der Reaktor

[1] Siehe z.B. [Pirt75]

während der Kultivierung besteht. Des weiteren soll jedoch auch den philosophisch-logischen Anforderungen entsprochen werden, welche nach [Sea96] für die Beschreibung eines *artifacts* notwendig sind.

Die Module sind im einzelnen:

Umgebung Der Reaktor befindet sich in einer Umgebung, welche Einfluß auf die Kultivierung ausüben kann.

Medium Das Kulturmedium hat eine eigenständige Funktion und ist auch physikalisch klar abgegrenzt von allen anderen Komponenten der Kultivierung.

Bioreaktor Der Bioreaktor dient zur Einflußnahme auf das Medium.

Kontrolle Letztlich wird der gesamte Vorgang von einer Steuereinheit überwacht und geregelt.

Echtzeit Die Kultivierung basiert auf der Einhaltung zeitlicher Bedingungen, daher ist eine Behandlung von Echtzeit in der Spezifikation dringend erforderlich.

Kultivierung Zur Beschreibung eines *artifacts* gehört zusätzlich zu der Beschreibung der physikalischen Eigenschaften auch die Beschreibung des Zwecks, welcher mit dem *artifact* verbunden ist. Im Modul *Kultivierung* wird dieser Zweck beschrieben und gehört zusammen mit der Spezifikation des Reaktors und der Kultivierung zum *artifact* Bioreaktor.

4.3 Zustandsvariablen und Konstanten

Zur Beschreibung der Kultivierung und des Reaktors wird in den verschiedenen Modulen eine große Anzahl an Zustandsvariablen und Konstanten deklariert. Tabelle C.3 auf Seite 70 gibt eine Übersicht darüber, welche Namen und Bedeutungen die einzelnen Zustandsvariablen haben und in welchen Modulen sie verwendet werden. In den Tabellen C.4 auf Seite 71 und C.5 auf Seite 72 sind die Konstanten der einzelnen Module aufgeführt.

4.4 Modul MiscDef

Siehe Anhang A.1 auf Seite 48

Dieses Modul dient zur Einbindung der Module *Reals*, *Naturals* und *Sequences*. Letzteres ist aus [Lad95] und wird dort zur Handhabung von Sequenzen benutzt. Die eingebundenen Module *Reals* und *Naturals* enthalten die üblichen algebraischen Operatoren für die Natürlichen und Reellen Zahlen. Die gebräuchlichen Definitionen zur Bestimmung von Absolutwert, Minimum und Maximum werden definiert.

$$
\begin{array}{lcl}
|z| & \triangleq & \textbf{if } z < 0 \textbf{ then } -z \\
 & & \textbf{else } z \\
min(a,b) & \triangleq & \textbf{if } a \leq b \textbf{ then } a \\
 & & \textbf{else } b \\
max(a,b) & \triangleq & \textbf{if } a \geq b \textbf{ then } a \\
 & & \textbf{else } b
\end{array}
$$

Das Modul dient zur Benutzung der Definitionen in anderen Modulen mittels **import** .

4.5 Modul Echtzeit

Siehe Anhang A.2 auf Seite 49

Entsprechend den Vorschlägen in [AL92] und [Lam93] werden die Definitionen für die Behandlung von Echtzeit bei Hybriden Systemen verwendet.

$$
\begin{array}{lcl}
RT(v) & \triangleq & \wedge\ now \in Real \\
 & & \wedge\ \Box[\wedge\ now' \in \{r \in Real : r > now\} \\
 & & \quad \wedge\ v' = v]_{now} \\
 & & \wedge\ \forall\, t \in Real : \Diamond(now > t)
\end{array}
$$

Dieser Ansatz zur Beschreibung von Echtzeit beruht auf der Einführung einer Zustandsvariablen *now*, welche den jeweiligen *Jetzt-Zeitpunkt* im beschriebenen System repräsentiert. Für diese Zustandsvariable gilt, daß sie nicht *rückwärts laufen* kann, sowie die Nonzeno-Bedingung. Von besonderer Bedeutung ist die Aussage, daß Zustandsübergänge des Systems nur *plötzlich* geschehen können. Während des Zustandsüberganges bleibt *now* unverändert. Dies bedeutet nicht, daß Veränderungen am System ohne Zeitverbrauch stattfinden. Es wird auf diese Weise lediglich eine Entkoppelung von Zeitverlauf und Systemverlauf ausgedrückt.

Um Zustandsveränderungen zeitlichen Beschränkungen zu unterwerfen, werden von Abadi und Lamport Timer definiert, die auch hier benutzt

werden.

$$
\begin{aligned}
VTimer(x, A, \delta, v) \;\triangleq\; & \wedge\; x = \textbf{if } \textsc{Enabled}\ \langle A \rangle_v \textbf{ then } now + \delta \\
& \qquad\qquad\qquad\qquad\qquad\qquad \textbf{else } \infty \\
& \wedge\; \Box[x' = \textbf{if } (\ \textsc{Enabled}\ \langle A \rangle_v)' \textbf{ then if } \langle A \rangle_v \textbf{ then } now' + \delta \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{else } x \\
& \qquad\qquad\qquad\qquad\qquad\qquad \textbf{else } \infty\ ]_{\langle x, v \rangle}
\end{aligned}
$$

Mittels $VTimer$ wird ein Timer x mit einer Action A, welche die Zustandsvariablen v verändert, und einem Zeitintervall δ verbunden. In allen Zuständen gilt, sobald $\textsc{Enabled}\ \langle A \rangle_v$ erfüllt ist, erhält x den Wert $now + \delta$, bis $\langle A \rangle_v$ eintritt. Ist $\textsc{Enabled}\ \langle A \rangle_v$ nicht erfüllt, so ist $x = \infty$.

$$MaxTimer(x) \;\triangleq\; \Box[(x \neq \infty) \Rightarrow (now' \leq x)]_{now}$$

$MaxTimer$ besagt, daß der Timer x von now nie überschritten wird.

$$ExactTimer(x, A, v) \;\triangleq\; \Box[\langle A \rangle_v \Rightarrow (now = x)]_v$$

$ExactTimer$ besagt, daß die Action $\langle A \rangle_v$ genau zum Zeitpunkt x eintritt. Diese Definition basiert auf der Definition für $MinTimer$ von Abadi/Lamport.

Für die Veränderung von Zustandsvariablen des Bioreaktors läßt sich zum Teil nur die Aussage treffen, daß sie sich verändern. Diese Veränderungen können zwar zu beliebigen Zeitpunkten geschehen, aber nur in einer kontinuierlichen Weise. Es ist daher notwendig, für die meisten Zustandsvariablen in den Modulen eine kontinuierliche Veränderung zu fordern.
Der Ausdruck

$$
\begin{aligned}
kontinuierlich(v) \;\triangleq\; & \forall\, i \in 1..Len(v) : \\
& \Box[\forall\, \delta \in Real^+ : \exists\, \epsilon \in Real^+ : now' - now < \epsilon \Rightarrow |v[i]' - v[i]| < \delta]_{v[i]}
\end{aligned}
$$

besagt für alle Zustandsvariablen aus der Folge v, daß in jedem Zustand jede Variable aus v entweder unverändert bleibt, oder der neue Wert konnte nur erreicht werden, wenn auch alle dazwischen liegenden Werte zu einem früheren Zeitpunkt erreicht wurden. Hiermit ist für die mit $kontinuierlich(v)$ benannten Zustandsvariablen in v keine sprunghafte Veränderung mehr möglich.

4.6 Modul Umgebung

Siehe Anhang A.3 auf Seite 50

Der Kultivierungsvorgang stellt an die Umgebung nur wenige Anforderungen. Zum einen soll die Umgebungstemperatur zwischen 10° C und 35° C liegen. Zum anderen darf die Lichteinwirkung ein bestimmtes Maß nicht überschreiten.

$$
\begin{array}{lcl}
tempBed & \triangleq & 10°C \leq temperatur \leq 35°C \\
lichtBed & \triangleq & helligkeit \leq MaxHelligkeit \\
vars & \triangleq & \langle temperatur, helligkeit \rangle \\
spec & \triangleq & \wedge\ \Box(tempBed \wedge lichtBed) \\
 & & \wedge\ RT(vars) \\
 & & \wedge\ kontinuierlich(vars)
\end{array}
$$

Für die benutzten Zustandsvariablen wird sowohl die Echtzeitbedingung RT als auch ein kontinuierliches Verhalten gefordert. Diese Bedingungen sind zwar für diese Spezifikation entbehrlich, gehören aber dennoch zur Beschreibung der Eigenschaften der Zustandsvariablen und könnten möglicherweise eine Rolle in anderen Spezifikationen spielen.

4.7 Modul Medium

Siehe Anhang A.4 auf Seite 51

Das Modul *Medium* beschreibt alle die Kultivierung betreffenden Zustandsveränderungen, welche das Kulturmedium vollziehen kann. Zunächst werden die Zustandsvariablen definiert, hierbei beschreibt

$$medium : Zutaten \cup Gifte \rightarrow Real^{0^+} \text{ VARIABLE}$$

daß das *medium* eine Funktion aus der Menge der Zutaten (Nährstoffe etc.) vereinigt mit der Menge der Gifte (Ausscheidungen) in die Reellen Zahlen ist. Der Zustand des Mediums ist damit durch eine Zuordnung von Inhaltsstoffen zu ihrer jeweiligen Menge, mit welcher dieser Stoff im Medium vorliegt, beschrieben.
Die Initialwerte der Zustandsvariablen als auch die Menge der Zutaten und Gifte sind als Konstanten definiert.

Es wird das Modul *Echtzeit* importiert. Da die Umgebung auf das Medium

Einfluß nimmt, wird auch dieses Modul eingebunden. Um eine Unterscheidung zwischen Umgebungstemperatur und Mediumstemperatur zu erreichen, findet eine Namenssubstitution statt.

include *Umgebung* **as** *umwelt* **with** $raumtemperatur \leftarrow temperatur$

Bei der Initialisierung der Zustandsvariablen wird angenommen, daß sich keine Bakterien im Medium befinden. Dies wird bei der tatsächlichen Kultivierung durch einen ca. fünftägigen Vorlauf mit Kulturmedium überprüft[2].

Folgende Veränderungen sind am Medium möglich:

$$wachstum \triangleq aktiveZellen' = aktiveZellen + 1$$

Durch Teilung kann genau eine neue Zelle entstehen.

$$\begin{aligned} sterben \triangleq\ & \wedge\ aktiveZellen > 0 \\ & \wedge\ aktiveZellen' = aktiveZellen - 1 \\ & \wedge\ toteZellen' = toteZellen + 1 \end{aligned}$$

Solange sich noch lebende Zellen im Medium befinden, kann eine Zelle absterben. Die Anzahl der lebenden Zellen verringert sich um eine Zelle und die Anzahl der toten erhöht sich entsprechend.

$$\begin{aligned} verbrauch \triangleq\ & \wedge\ \exists\, z \in Zutaten\ :\ 0 \leq medium'[z] < medium[z] \\ & \wedge\ \forall\, x \in Zutaten \cup Gifte\ :\ (x = z \vee \text{UNCHANGED}\ medium[x]) \end{aligned}$$

besagt, daß zumindest eine Zutat im Medium aufgrund einer Aufnahme durch die Zelle nach dem Verbrauch mit einer geringeren Menge im Kulturmedium vorliegt. Um den Verbrauch der verschiedenen Inhaltsstoffe voneinander unabhängig zu machen, bleiben die übrigen Inhaltsstoffe unverändert.

$$\begin{aligned} ausscheidung \triangleq\ & \wedge\ \exists\, g \in Gifte\ :\ medium'[g] > medium[g] \\ & \wedge\ \forall\, x \in Zutaten \cup Gifte\ :\ (x = g \vee \text{UNCHANGED}\ medium[x]) \end{aligned}$$

beschreibt auf die selbe Weise wie *verbrauch* die Vermehrung der Ausscheidungen im Kulturmedium.

[2] vgl. auch Absatz 2.3.2 auf Seite 6

$$pHaenderung \triangleq \begin{array}{l} \wedge\ 0 \leq pH' \leq 14 \\ \wedge\ pH' \neq pH \end{array}$$

beschreibt eine beliebige Veränderung des pH zwischen 0 und 14.

$$pO_2aenderung \triangleq \begin{array}{l} \wedge\ 0 \leq pO_2' \leq 460 \\ \wedge\ pO_2' \neq pO_2 \end{array}$$

besagt ebendieses für den pO_2 Wert zwischen den Werten 0 bis 460. Hierbei ist 100 der der Luft entsprechend pO_2 Wert; ein Wert von 460 entspricht damit fast reinem O_2.[3]

$$abkuehlung \triangleq raumtemperatur \leq temperatur' < temperatur$$

Das Medium kann sich maximal bis auf Umgebungstemperatur abkühlen.

$$baktWachstum \triangleq \begin{array}{l} \wedge\ bakterien > 0 \\ \wedge\ bakterien' > bakterien \end{array}$$

Falls Bakterien in das Medium gelangt sind, so können sie sich beliebig vermehren.

Alle Definitionen im Abschnitt *actions* beinhalten zusätzlich eine UNCHANGED Anweisung für sämtliche Zustandsvariablen, welche nicht von der jeweiligen Action direkt verändert werden. Dies ist notwendig, um zum einen ein unkontrolliertes Verändern dieser Werte zu verhindern. Zum anderen würde das Weglassen der UNCHANGED Anweisung eine in der Realität nicht gegebene und damit unzulässige Verknüpfung der Wertveränderung mit der jeweiligen Action bedeuten.

Für die Beschreibung der temporalen Bedingungen werden die kontinuierlichen Zustandsvariablen als $vars1$ zusammengefaßt.

$$vars1 \triangleq \langle pH, pO_2, temperatur, raumtemperatur \rangle$$
$$vars \triangleq vars1 \circ \langle medium, aktiveZellen, toteZellen, bakterien \rangle$$

Die Actions werden disjunkt unter dem Namen *actions* verknüpft.

[3] Erläuterung in Fußnote 1 auf Seite 38

$$
\begin{aligned}
spec \triangleq\ & \wedge\ Init \\
& \wedge\ \Box[actions]_{vars} \\
& \wedge\ RT(vars) \\
& \wedge\ kontinuierlich(vars1)
\end{aligned}
$$

Die Spezifikation *spec* konjugiert die Initialisierung der Zustandsvariablen mit der Aussage, daß eine Veränderung der Zustandsvariablen nur über die beschriebenen *actions* geschehen kann. Weiterhin unterliegen alle Zustandsvariablen der Echtzeit-Bedingung. Für die kontinuierlich verlaufenden Zustandsvariablen wird ein entsprechender Verlauf beschrieben.

4.8 Modul Bioreaktor

Siehe Anhang A.5 auf Seite 53

Dieses Modul beschreibt, welche Möglichkeiten der Bioreaktor bietet, gezielt auf das Medium von Außen einzuwirken.

Nach der Deklaration der Zustandsvariablen und Konstanten wird das Modul *Echtzeit* importiert.

import *Echtzeit*
include *Medium* **as** *medium*

Da sich das Medium im Bioreaktor befindet, ist es notwendig, das Modul *Medium* einzubinden. Die Zustandsvariablen werden unverändert in das Modul *Bioreaktor* übernommen.

Die Initialisierung der Zustandsvariablen wird unter der Bezeichnung *Init* definiert.

Nachfolgend sind jene Actions aufgelistet, welche eine Veränderung des Mediums unmittelbar am Reaktor zulassen.

$$
\begin{aligned}
neueZellen(a,t) \triangleq\ & \wedge\ a \in Nat^{+} \\
& \wedge\ t \in Nat^{0+} \\
& \wedge\ aktiveZellen' = aktiveZellen + a \\
& \wedge\ toteZellen' = toteZellen + t \\
& \wedge\ volumen < volumen' \leq Maxvolumen
\end{aligned}
$$

Es können sowohl neue lebende als auch tote Zellen in das Kulturmedium

gegeben werden. Die jeweilige Anzahl der Zellen im Medium erhöht sich um die entsprechende Anzahl a oder t. Das Volumen des Mediums vergrößert sich, bleibt aber unter dem vorgegebenen Maximum, in diesem Fall 20 Liter.

$$\begin{aligned} neuesMedium(m) \triangleq\ & \wedge\ m \in Zutaten \to Real^{0+} \\ & \wedge\ \forall\, x \in Zutaten\ :\ medium[x]' = medium[x] + m[x] \\ & \wedge\ volumen < volumen' \leq Maxvolumen \end{aligned}$$

Es ist möglich, Medium nachzufüllen. Hierbei wird die Zusammensetzung des einzufüllenden Mediums auf die gleiche Weise beschrieben wie die Zusammensetzung des Mediums im Reaktor. Zu jeder Zutat ist angegeben, in welcher Menge sie in der Zugabe enthalten ist. Zutaten, die nicht nachgefüllt werden, erhalten den Funktionswert 0. Die Menge jeder einzelnen Zutat im Kulturmedium nach der Befüllung ergibt sich daher aus der Menge, die vor der Befüllung im Reaktor war, plus der Menge aus der Zugabe.

$$\begin{aligned} entnahme(a,t,m) \triangleq\ & \wedge\ t, a \in Nat^{0+} \\ & \wedge\ a \leq aktiveZellen \\ & \wedge\ t \leq toteZellen \\ & \wedge\ m \in Zutaten \to Real^{+} \\ & \wedge\ aktiveZellen' = aktiveZellen - a \\ & \wedge\ toteZellen' = toteZellen - t \\ & \wedge\ \forall\, x \in Zutaten\ :\ \wedge\ medium[x]' = medium[x] - m[x] \\ & \qquad\qquad\qquad\qquad\ \wedge\ 0 \leq medium[x]' \\ & \wedge\ 0 \leq volumen' < volumen \end{aligned}$$

Ebenso besteht die Möglichkeit, Kulturmedium zu entnehmen. Dies kann sowohl mit Zellmaterial als auch ohne geschehen. Die Entnahme ist beschränkt auf die Menge von Inhaltsstoffen und Zellen, die sich noch im Reaktor befindet. Die entnommene Menge Medium sowie lebender und toter Zellen wird von der aktuellen Anzahl im Reaktor abgezogen. Das Volumen des Kulturmediums verringert sich dadurch.

$$\begin{aligned} verseuchung(n) \triangleq\ & \wedge\ n \in Nat^{+} \\ & \wedge\ bakterien' = bakterien + n \end{aligned}$$

Beim Einbringen oder Entnehmen von Medium oder Zellen ist es möglich, daß eine Verseuchung des Kulturmediums mit Bakterien stattfindet.
Dies sind alle Möglichkeiten, direkt über den Reaktor auf das Medium einzuwirken.

Die weiteren Einwirkungsmöglichkeiten können nur über die Kontrolleinheit am Reaktor vorgenommern werden. Sie werden zwar im Modul *Bioreaktor* definiert, gehören aber nicht zur Spezifikation dieses Moduls, sondern zur Spezifikation des Moduls *Kontrolle*.
Diese sind:

$$\begin{aligned} heizen(t) \triangleq\ & \wedge\ t > temperatur \\ & \wedge\ temperatur' = t \end{aligned}$$

Die Temperatur des Mediums kann über die Heizung erhöht werden.

$$\begin{aligned} drehzahl(d) \triangleq\ & \wedge\ d \in Real^{0+} \\ & \wedge\ d \neq ruehrerdrehzahl \\ & \wedge\ ruehrerdrehzahl' = d \end{aligned}$$

Die Drehzahl des Rührers wird auf einen neuen Wert eingestellt.

$$\begin{aligned} mGas(co, o, n, l) \triangleq\ & \wedge\ co, o, n, l \in Real^{0+} \\ & \wedge\ co + o + n + l = 100 \\ & \wedge\ \langle co, o, n, l \rangle \neq membrangas \\ & \wedge\ membrangas' = \langle co, o, n, l \rangle \end{aligned}$$

Die prozentuale Zusammensetzung der Gasmischung in der Begasungsmembran wird neu eingestellt.

$$\begin{aligned} kGas(wahl) \triangleq\ & \wedge\ wahl \in \{\text{“luft”}, \text{“gemisch”}\} \\ & \wedge\ kopfraumgas' = \textbf{if}\ wahl = \text{“luft”}\ \textbf{then}\ \langle 0, 0, 0, 100 \rangle \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{else}\ membrangas' \end{aligned}$$

Bei der Begasung des Kopfraumes kann gewählt werden zwischen dem Gasgemisch, welches auch in der Begasungsmembran ist, oder Luft.

$$\begin{aligned} mFluss(m) \triangleq\ & \wedge\ m \in Real^{0+} \\ & \wedge\ m \neq membrangasfluss \\ & \wedge\ membrangasfluss' = m \end{aligned}$$

$$kFluss(m) \triangleq \begin{array}{l} \wedge\ m \in Real^{0+} \\ \wedge\ m \neq kopfraumgasfluss \\ \wedge\ kopfraumgasfluss' = m \end{array}$$

$$mDruck(d) \triangleq \begin{array}{l} \wedge\ d \in Real^{+} \\ \wedge\ d \neq membrandruck \\ \wedge\ membrandruck' = d \end{array}$$

$$kDruck(d) \triangleq \begin{array}{l} \wedge\ d \in Real^{+} \\ \wedge\ d \neq kopfraumdruck \\ \wedge\ kopfraumdruck' = d \end{array}$$

Die Veränderung der Massendurchflüsse und des jeweiligen Druckes in der Begasungsmembran und im Kopfraum lassen sich mittels o.g. Actions neu einstellen.

Wie schon im Modul *Medium* sind auch hier alle Actions mit UNCHANGED Anweisungen versehen, die sich auf die unveränderten Zustandsvariablen beziehen.

Für die Beschreibung der temporalen Bedingungen werden die kontinuierlichen Zustandsvariablen als $vars1$ zusammengefasst.

$$\begin{array}{l} vars1 \triangleq \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, membrangas, membrangasfluss, \\ \qquad kopfraumgas, kopfraumgasfluss, temperatur, volumen \rangle \\ vars \triangleq vars1 \circ \langle medium, aktiveZellen, toteZellen, bakterien \rangle \end{array}$$

$$actions \triangleq \begin{array}{l} \vee\ \exists\, a, t : neueZellen(a, t) \\ \vee\ \exists\, m : neuesMedium(m) \\ \vee\ \exists\, a, t, m : entnahme(a, t, m) \\ \vee\ \exists\, n : verseuchung(n) \end{array}$$

Nur die Actions, welche direkt am Bioreaktor vorgenommen werden können, werden disjunkt unter dem Namen *actions* verknüpft.

$$spec \triangleq \begin{array}{l} \wedge\ Init \\ \wedge\ \Box[actions]_{vars} \\ \wedge\ RT(vars) \\ \wedge\ kontinuierlich(vars1) \end{array}$$

In der Spezifikation *spec* wird auf gleiche Weise wie bei den vorigen Modulen die Initialisierung mit der Sicherheitsbedingung für die Veränderung der Zustandsvariablen, der Echtzeitbedingung und der Einschränkung für die kontinuierlichen Zustandsvariablen konjugiert. Auf diese Weise ist beschrieben, welche Veränderungen unmittelbar durch den Bioreaktor am Medium stattfinden dürfen.

4.9 Modul Kontrolle

Siehe Anhang A.6 auf Seite 57

Das Modul *Kontrolle* beschreibt, welche Anforderungen an die Überwachung der Kultivierung gestellt werden. Dies ist in erster Linie die Einhaltung der zeitlichen Beschränkungen für die Abweichung von den Sollwerten. Es handelt sich hierbei nicht um die Beschreibung einer speziellen Regelungsmethode, wie z.B. der PID-Regelung, sondern um die Beschreibung von Anforderungen, welche jede Regelungsmethode erfüllen muß. Diese Anforderungen ergeben sich aus den Bedingungen, innerhalb derer die Kultivierung möglich ist. Dies ermöglicht es, jeden beliebigen Regelungsmechanismus auf die Erfüllung der hier genannten Regelungsbedingungen zu überprüfen.

Die Zustandsvariablen aus dem Modul *Bioreaktor*, welches unter dem Namen *reaktor* eingebunden wird, behalten ihre ursprünglichen Namen. Lediglich die Zustandsvariablen für die Sollwerte stammen direkt aus dem Modul *Kontrolle*.

Der üblichen Initialisierung folgt die Beschreibung der Actions.

$$\begin{aligned}
pH_soll(s) &\triangleq \land\ 0 \leq s \leq 14 \\
&\quad\ \land\ pH^{s'} = s \\
pO_2_soll(s) &\triangleq \land\ 0 \leq s \leq 460 \\
&\quad\ \land\ pO_2^{s'} = s \\
temp_soll(s) &\triangleq \land\ 10 \leq s \leq 80 \\
&\quad\ \land\ temp^{s'} = s \\
drehz_soll(s) &\triangleq \land\ 0 \leq s \leq 50 \\
&\quad\ \land\ drehz^{s'} = s \\
mDruck_soll(s) &\triangleq \land\ 0 \leq s \\
&\quad\ \land\ mDruck^{s'} = s
\end{aligned}$$

$$
\begin{array}{lcl}
kDruck\_soll(s) & \triangleq & \wedge\ 0 \leq s \\
 & & \wedge\ kDruckd^{s\prime} = s \\
mFluss\_soll(s) & \triangleq & \wedge\ 0 \leq s \\
 & & \wedge\ mFluss^{s\prime} = s \\
kFluss\_soll(s) & \triangleq & \wedge\ 0 \leq s \\
 & & \wedge\ kFluss^{s\prime} = s
\end{array}
$$

Die Einstellung neuer Sollwerte geschieht direkt über die Zuweisung eines neuen Wertes an die jeweilige Zustandsvariable.

Von besonderer Bedeutung ist der Umstand, daß eine Korrektur des pH oder pO_2 Wertes nur indirekt über eine geeignete Einstellung der Begasung erreicht werden kann. Dies unterscheidet sie von den anderen Regelwerten Temperatur, Rührerdrehzahl, Druck und Massendurchfluß.

$$
\begin{array}{lcl}
pH\_korrektur & \triangleq & \wedge\ pH \neq pH^s \\
 & & \wedge\ \exists\, co, o, n, l\ :\ reaktor.mGas(co, o, n, l) \\
 & & \wedge\ \exists\, w\ :\ reaktor.kGas(w) \\
 & & \wedge\ pH' = pH^s \\
pO_2\_korrektur & \triangleq & \wedge\ pO_2 \neq pO_2^s \\
 & & \wedge\ \exists\, co, o, n, l\ :\ reaktor.mGas(co, o, n, l) \\
 & & \wedge\ \exists\, w\ :\ reaktor.kGas(w) \\
 & & \wedge\ pO_2' = pO_2^s
\end{array}
$$

Die Voraussetzung für eine Korrektur ist, daß der momentane Wert vom Sollwert abweicht. Ist dies der Fall, so soll über die Einstellung der Gaszusammensetzung und der Begasungsart des Kopfraumes der Sollwert wieder erreicht werden.

Auch hier beinhalten alle Actions eine UNCHANGED Anweisung, um alle unveränderten Zustandsvariablen unter Kontrolle zu behalten.

Zur Spezifikation der Reglung muß beschrieben werden, daß der aktuelle Wert nicht länger als eine maximale Zeitspanne vom geforderten Sollwert abweicht. Dies wird mittels der im Modul *Echtzeit* eingeführten Timer realisiert.

$$
\begin{array}{lcl}
pH\_richtig & \triangleq & \exists\, t\ :\ \wedge\ VTimer(t, pH\_korrektur, \Delta_{pH}, pH) \\
 & & \qquad\ \wedge\ MaxTimer(t)
\end{array}
$$

$$pO_2\_richtig \triangleq \exists\, t : \wedge\, VTimer(t, pO_2\_korrektur, \Delta_{pO_2}, pO_2) \wedge\, MaxTimer(t)$$

$$temp\_richtig \triangleq \exists\, t : \wedge\, VTimer(t, reaktor.heizen(temp^s), \Delta_{temp}, temperatur) \wedge\, MaxTimer(t)$$

$$drehz\_richtig \triangleq \exists\, t : \wedge\, VTimer(t, reaktor.drehzahl(drehz^s), \Delta_{drehz}, ruehrerdrehzahl) \wedge\, MaxTimer(t)$$

$$mDruck\_richtig \triangleq \exists\, t : \wedge\, VTimer(t, reaktor.mDruck(mDruck^s), \Delta_{mDruck}, membrandruck) \wedge\, MaxTimer(t)$$

$$kDruck\_richtig \triangleq \exists\, t : \wedge\, VTimer(t, reaktor.kDruck(kDruckd^s), \Delta_{kDruck}, kopfraumdruck) \wedge\, MaxTimer(t)$$

$$mFluss\_richtig \triangleq \exists\, t : \wedge\, VTimer(t, reaktor.mFluss(mFluss^s), \Delta_{mFluss}, membranfluss) \wedge\, MaxTimer(t)$$

$$kFluss\_richtig \triangleq \exists\, t : \wedge\, VTimer(t, reaktor.kFluss(kFluss^s), \Delta_{kFluss}, kopfraumfluss) \wedge\, MaxTimer(t)$$

Zunächst wird eine neue Zustandsvariable t instanziiert, welche als Timer dient. Mittels *VTimer* benennt dieses t den Zeitpunkt, zu welchem eine Korrektur auf den Sollwert spätestens stattfinden sollte, um das Zeitintervall Δ nicht zu überschreiten. Die Forderung, daß t ein *MaxTimer* ist, sorgt dafür, daß der Zeitpunkt tatsächlich nicht überschritten wird. Die beiden Timerbedingungen ergeben zusammen, daß nach Verlassen des Sollwertes dieser innerhalb der Zeitspanne Δ wieder erreicht wird. Hiermit ist beschrieben, welcher Anforderung jeder Regelungsmechanismus genügen muß.

Neben der Einhaltung bestimmter Zeitintervalle ist es für die erfolgreiche Durchführung der Kultivierung notwendig, daß

- die Luftsättigung
- der *pH*
- die Mediumstemperatur
- und die Rührerdrehzahl

niemals Werte annehmen, die zur Zerstörung der Zellkultur führen könnten. Dieses wird in

$$\begin{aligned} grenzBedingung \triangleq\ & \wedge\, Min\_pO_2 \leq pO_2 \leq Max\_pO_2 \\ & \wedge\, Min\_pH \leq pH \leq Max\_pH \\ & \wedge\, Min\_temp \leq temperatur \leq Max\_temp \\ & \wedge\, Min\_drehz \leq ruehrerdrehzahl \leq Max\_drehz \end{aligned}$$

beschrieben.

Die *actions* des Moduls sind die Veränderungen der Sollwerte. Andere Veränderungsmöglichkeiten, welche die Kultivierung betreffen, können an der Steuereinheit nicht vorgenommen werden.

Die Spezifikation besteht aus der Konjunktion der Initialisierung, den Timer-Aussagen, der Grenzwertbedingung für alle Zustände und dem Sicherheitsausdruck für die Actions, welche die Sollwerte verändern.

$$
\begin{aligned}
spec \triangleq\ & \wedge\ Init \\
& \wedge\ pH\_richtig \\
& \wedge\ pO_2\_richtig \\
& \wedge\ temp\_richtig \\
& \wedge\ drehz\_richtig \\
& \wedge\ mDruck\_richtig \\
& \wedge\ kDruck\_richtig \\
& \wedge\ mFluss\_richtig \\
& \wedge\ kFluss\_richtig \\
& \wedge\ \Box grenzBedingung \\
& \wedge\ \Box[actions]_{sollVars}
\end{aligned}
$$

Die in den anderen Modulen in der Spezifikation benutzten Ausdrücke für Echtzeit oder kontinuierliche Variablen werden hier nicht gebraucht, da diese Forderungen für die Zustandsvariablen aus dem Modul *Bioreaktor* bereits in der dortigen Spezifikation gestellt wurden. Für die Zustandsvariablen aus diesem Modul, den Sollwerten, können diese Bedingungen nicht gefordert werden, da sie weder ein Echtzeit- noch ein kontinuierliches Verhalten zeigen. Zusammenfassend besagt die Spezifikation dieses Moduls, daß die Sollwerte einen initialen Wert besitzen, daß die Zeitintervalle und die Grenzbedingung eingehalten werden müssen und daß die Sollwerte unverändert bleiben, solange sie nicht von außen verändert werden.

4.10 Modul Kultivierung

Siehe Anhang A.7 auf Seite 61

Das Modul *Kultivierung* verbindet die Spezifikationen der einzelnen Module und beschreibt auf diese Weise den Bioreaktor zusammen mit seiner Umgebung, dem enthaltenen Medium und den Anforderungen an die

Regelung. Zusätzlich wird das Ziel des Kultivierungsprozesses beschrieben.

Nach der üblichen Deklaration der Variablen und Konstanten erfolgt das Einbinden aller Module.

import *MiscDef*
import *Echtzeit*
include *Umgebung* **as** *umgebung*
include *Medium* **as** *medium*
include *Bioreaktor* **as** *bioreaktor*
include *Kontrolle* **as** *kontrolle*

Zur Beschreibung des Ziels ist es notwendig, die Verdoppelung der Zellzahl innerhalb der Verdoppelungszeit zu beschreiben. Zu diesem Zweck gibt es eine Action *verdoppelung*.

$$\begin{aligned} \mathit{verdoppelung} \triangleq\ & \wedge\ \mathit{aktiveZellen} \geq 2 \cdot \mathit{letzte} \\ & \wedge\ \mathit{letzte}' = \mathit{aktiveZellen} \\ & \wedge\ \text{UNCHANGED}\ \langle \mathit{bakterien}, \mathit{volumen} \rangle \end{aligned}$$

Eine Verdoppelung liegt immer dann vor, wenn sich seit der letzten Verdoppelung die Anzahl an lebenden Zellen erneut verdoppelt hat. In diesem Fall wird die Anzahl der lebenden Zellen *letzte* zugewiesen. Diese Action findet nun Anwendung in der Beschreibung des Ziels.

$$\begin{aligned} \mathit{ziel} \triangleq\ & \wedge\ \exists\, x : \wedge\ \mathit{Vtimer}(x, \mathit{verdoppelung}, \Delta_{\mathit{doppel}}, \mathit{letzte}) \\ & \qquad\ \wedge\ \mathit{MaxTimer}(x) \\ & \wedge\ \Box(\vee\ \mathit{aktiveZellen} \geq \mathit{AktiveZellen}_{\mathit{max}} \\ & \qquad \vee\ \mathit{bakterien} \geq \mathit{SichtAnzahl}) \Rightarrow (\wedge\ \exists\, m, a, t : \mathit{bioreaktor.entnahme}(a, t, m) \\ & \qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \mathit{volumen}' = 0) \end{aligned}$$

Das Ziel der Kultivierung ist es, innerhalb der Verdoppelungszeit Δ_{doppel} eine Verdoppelung der Kultur zu erreichen, bis eine maximale Anzahl an Zellen erreicht wurde oder eine Verseuchung mit Bakterien sichtbar wird. In beiden Fällen führt dies zum Entleeren des Reaktors. Daher ist das Ziel beschrieben als die Konjunktion der Timer Bedingungen für die Verdoppelung mit der folgenden Aussage: In jedem Zustand führt entweder das Erreichen der maximalen Zellzahl oder eine sichtbare Verseuchung mit Bakterien zu einer vollständige Entleerung des Reaktors.

Leider kann mit dem vorhandenen Bioreaktor nicht garantiert werden, daß dieses Ziel auch tatsächlich immer erreicht wird, selbst wenn alle in dieser Spezifikation geforderten Bedingungen und Anforderungen erfüllt

werden. Die Spezifikation des Gesamtsystems beinhaltet daher nur die Spezifikationen der anderen Module, nicht aber auch das Ziel.

$$
\begin{aligned}
spec \triangleq\ & \wedge\ Init \\
& \wedge\ umgebung.spec \\
& \wedge\ medium.spec \\
& \wedge\ bioreaktor.spec \\
& \wedge\ kontrolle.spec
\end{aligned}
$$

Welche Bedeutung die einzelnen Spezifikationen haben und was somit die Gesamtspezifikation beschreibt, ergibt sich aus der folgenden Zusammenfassung.

umgebung.spec Die Umgebungstemperatur bewegt sich nur zwischen 10° C und 35° C. Die Lichteinstrahlung in das Medium übersteigt einen maximalen Wert nicht. Die Zustandsvariablen unterliegen den Bedingungen für Echtzeit und Kontinuität.

medium.spec Das Kulturmedium bleibt unverändert, solange nicht eine der folgenden Veränderungen auftritt:

- Es enstehen neue Zellen
- Es sterben Zellen
- Nährstoffe werden verbraucht
- Ausscheidungen gelangen in das Kulturmedium
- Der pH verändert sich beliebig
- Die Luftsättigung verändert sich beliebig
- Das Kulturmedium kühlt sich ab
- Vorhandene Bakterien vermehren sich

Alle Zustandsvariablen unterliegen den Bedingungen für Echtzeit, und für einige gilt, daß sie sich nur kontinuierlich verändern können.

bioreaktor.spec Das Kulturmedium bleibt unverändert, solange nicht eine der folgenden Veränderungen auftritt:

- Neue lebende oder tote Zellen gelangen in das Kulturmedium
- Neues Basalmedium gelangt in das Kulturmedium
- Kulturmedium wird entnommen

- Bakterien gelangen in das Kulturmedium

Alle Zustandsvariablen unterliegen den Bedingungen für Echtzeit, und für einige gilt, daß sie sich nur kontinuierlich verändern können.

kontrolle.spec Die Zustandsvariablen für

- pH, geregelt über die Membrangasmischung und die Kopfraumbegasung,
- pO_2, geregelt über die Membrangasmischung und die Kopfraumbegasung,
- Temperatur
- Rührerdrehzahl
- Druck in der Membran und im Kopfraum
- Massendurchfluß durch Membran und Kopfraum

weichen maximal die geforderte Zeit vom Sollwert ab und überschreiten niemals die geforderten Grenzwerte.
Solange die Sollwerte nicht neu eingestellt werden, bleiben diese unverändert.

Anhand der vorliegenden Beschreibung des Bioreaktors, des Mediums und der Bedingungen für die Regelung sowie der formalen Spezifikation des Ziels, besteht die Möglichkeit, Implementationen sowohl für die Regelung als auch für andere Aspekte des Reaktors oder der Kultivierung, wie z.B. der Automatisierung des Wechsels von Basalmedium, auf die Erfüllung der vorliegenden Spezifikation zu überprüfen.

Kapitel 5

Spezifikation der Regelungssoftware

Susanne Bockholt hat in [Bock96] die Entwicklung einer Softwaresteuerung für die Begasungsregelung dokumentiert und implementiert. Leider gibt es zu dieser Software keine Spezifikation bzw. Entwurfsplanung oder andere Aufzeichnungen zur Entwicklung. Lediglich im Quellcode findet sich eine rudimentäre Dokumentation der Programmzeilen. Außer einer einfachen Bedienungsanleitung gibt es keine weitere Dokumentation der Software. Die nachfolgend dargestellte Spezifikation dieser Software beruht daher auf dem Quelltext und den Informationen aus der Diplomarbeit sowie der Bedienungsanleitung. Hierbei war es notwendig, die Arbeitsweise der Software von dem implementationsnahen Niveau des Quelltextes auf ein höheres Abstraktionsniveau abzubilden. Einige Teile der Software sind nicht in einer TLA$^+$ Spezifikation beschrieben worden, zum einen, um den Umfang der Softwarespezifikation möglichst gering zu halten, zum anderen auf Grund fehlender Detailinformationen. Da es sich dennoch um wichtige Aspekte der Software handelt, gibt es fiktive Module, die in die vorhandenen Module eingebunden werden. Die Aufgabe der nicht beschriebenen Module ist lediglich die Berechnung von bestimmten Zahlenwerten. Ein Weglassen der Beschreibung schmälert daher nicht die Aussagekraft der Spezifikationen der vorhandenen Module. Die Implementation der Bedienungsoberfläche wurde bei der Softwarespezifikation vollständig weggelassen. Nur die Teile der Software, die direkt mit der Steuerung der Begasung in Zusammenhang stehen, sind in der nachfolgenden Spezifikation enthalten.

5.1 Softwareregelung vs. Analogen PID-Regler

Wie in Abschnitt 2.3.3 auf Seite 7 beschrieben, muß die analoge PID Regelungsanlage von Hand auf eine Begasung mit Luft/O_2 umgeschaltet werden. Dies erfordert eine ständige Überwachung des pO_2 Stellwertes durch das wissenschaftliche Personal, welche natürlich nicht gegeben ist. Erreicht der Stellwert z.B. in der Nacht den kritischen Wert, so kann eine Unterversorgung der Zellen mit Sauerstoff bis zum nächsten Morgen die Folge sein. Der Grund für die Handschaltung liegt im Mischungsvorgang des Gases durch die PID-Anlage.

Entsprechend dem pO_2 Stellwert bestimmt die Reglereinheit den Beimischwert an fremdem Gas (N_2 oder O_2) zur Luft. Mit steigendem Stellwert steigt dieser Beimischwert. Da aber bei der Luft/N_2 Begasung mit steigendem Stellwert der Anteil an N_2 abnehmen muß, wird bei dieser Begasungsarts der Beimischwert negiert. Zum Zeitpunkt der Umschaltung hat der Beimischwert sein Maximum erreicht, was auf Grund der Negierung zu einem minimalen Anteil an N_2 führt. Nach dem Umschalten wird der Beimischwert nun ohne Negierung für die Bestimmung des O_2 Anteils verwendet. Dies führt zu einer Begasung mit maximaler O_2 Menge direkt nach dem Umschalten. Der Beimischwert fällt nur langsam, sodaß eine Überschreitung des pO_2 Sollwertes nicht verhindert werden kann. Die Folge ist ein fallender pO_2 Stellwert. Durch die Umschaltung wird demnach zum einen die gesamte Regelung durcheinander gebracht, zum anderen ergibt sich ein fallender Stellwert, der erst nach einer gewissen Zeit wieder parallel mit dem Wachstum der Zellen anwächst.

Die Softwaresteuerung überwacht und vollzieht hingegen diese Umschaltung selbstständig. Es wird ein gleitender Übergang zwischen der Begasung Luft/N_2 und Luft/O_2 geschaffen. Es gibt somit keinen Einbruch im Verlauf des pO_2 Stellwertes, und es ergibt sich die Möglichkeit, anhand der Kurve des Stellwertes die Anzahl der Zellen im Kulturmedium zu bestimmen. Ein weiterer Vorteil der Software ist die Bedienungsoberfläche, welche dem Nutzer des Reaktors eine sehr komfortable Möglichkeit bietet, einzelne Stellwerte manuell zu steuern.

5.2 Zustandsvariablen und Konstanten

Ebenso wie bei den Modulen der Reaktorbeschreibung gibt es auch in den Modulen zur Softwarespezifikation eine große Anzahl an Zustandsvariablen

und Konstanten . Die Zustandsvariablen sind in der Tabelle C.1 auf Seite 68 aufgelistet. Die Konstanten finden sich in der Tabelle C.2 auf Seite 69.

5.3 Modul Sensorprozess

Siehe Anhang B.1 auf Seite 62

Die Reaktorsteuerung kommuniziert mit den Sensoren bzw. dem Analog/Digitalwandler über eine Prozeßkommunikationsstruktur, welche vom Betriebssystem zur Verfügung gestellt wird. Es gibt für jeden Sensor einen einzelnen Prozeß, welcher Anfragen auf Abruf beantwortet. Dieser Sensorprozeß wird in diesem Modul beschrieben.

Nach der Deklarierung der Zustandsvariablen werden u.a. die Module *Umrechnung*, *A/D_Wandler* und *Filter* eingebunden.

import *Umrechnung*
include *A/D_Wandler* **as** *ADWandler* **with** *output* $\leftarrow$ *digWert*
include *Filter* **as** *filter* **with** *filterWert* $\leftarrow$ *fBitWert*

Für diese Module gibt es keine TLA$^+$ Spezifikation, daher werden die aus diesen Modulen benutzten Definitionen und Actions hier kurz erläutert.

Das Modul *Umrechnung* beinhaltet die Definitionen *bitToVolt* und *bitToWert*. Sie berechnen aus einem digitalen Wert den entsprechenden Spannungswert bzw. den Wert in der jeweiligen physikalischen Einheit.

Das Modul *A/D_Wandler* beschreibt den A/D-Wandler und bietet die Action *neuerWert*, welche die Zustandsvariable *output* mit einem neuen Meßwert belegt.

Mittels des Moduls *Filter* wird über die Action *filter*(x) der Wert von x gefiltert und der Zustandsvariablen *filterWert* zugewiesen.

$$anpassung(x) \triangleq (x + Verschiebung) \cdot Steigung$$

Mittels der *anpassung* wird der Meßwert auf den Sensor geeicht.

$$
\begin{aligned}
wertNeu \triangleq\ & \wedge\ ADWandler.neuerWert(Kanal) \\
& \wedge\ bitWert' = min(max(MinBitWert, anpassung(digWert')), MaxBitWert) \\
& \wedge\ filter.filter(bitWert') \\
& \wedge\ fVolt' = bitToVolt(fBitWert') \\
& \wedge\ fWert' = bitToWert(fBitWert', Einheit) \\
& \wedge\ trigger' \neq trigger
\end{aligned}
$$

Die Action *wertNeu* sorgt für eine Aktualisierung des Meßwertes *digWert* auf dem angegebenen Kanal mittels der Action *neuerWert* aus dem Modul *AD_Wandler*. Dieser neue Meßwert wird durch Anwendung der *anpassung* geeicht und in das Intervall der zulässigen Werte eingepaßt. Mittels Filterung und Umrechnung werden die neuen gefilterten Werte für den digitalen Meßwert (*fBitWert*), die Sensorspannung (*fVolt*) und den Meßwert in der physikalischen Einheit (*fWert*) berechnet. Um ein zyklisches Erneuern der Meßwerte zu erreichen, wird die Zustandsvariable *trigger* verändert.

$$
\begin{aligned}
spec \triangleq\ & \wedge\ \exists\, x\ :\ \wedge\ Vtimer(x, wertNeu, \Delta, trigger) \\
& \qquad\qquad \wedge\ ExactTimer(x, wertNeu, trigger) \\
& \wedge\ \Box[wertNeu]_{\langle digWert, bitWert, fBitWert, fVolt, fWert, trigger \rangle}
\end{aligned}
$$

Die Spezifikation des Sensorprozesses besteht einerseits aus einem Timer x, der besagt, daß immer exakt nach der Zeitspanne Δ neue Meßwerte bestimmt werden. Andererseits ist es nur mittels der Action *wertNeu* möglich, die Zustandsvariablen der Meßwerte zu verändern.

5.4 Modul Aktorprozess

Siehe Anhang B.2 auf Seite 64

Entsprechend dem Sensorprozeß gibt es auch für jedes Stellglied einen eigenen Aktorprozeß. Auch in diesem Modul werden zwei nicht spezifizierte Module eingebunden.

import *Umrechnung*
include *D/A_Wandler* **as** *DAWandler*

Das Modul *Umrechnung* wurde im vorhergehenden Absatz 5.3 beschrieben. Die Umwandlung und Weiterleitung des digitalen Stellwertes an den Regler beschreibt das Modul *D/A_Wandler*.

Wie beim Modul *Sensorprozess* gibt es auch hier nur eine Action, die für die Neueinstellung der Stellwerte sorgt und die Weiterleitung dieser Werte veranlaßt.

$$\begin{aligned} putData \triangleq\ & \wedge\ bitWert' = wertToBit(wert, Kanal) \\ & \wedge\ volt' = bitToVolt(bitWert') \\ & \wedge\ DAWandler.DA\_Wandlung(bitWert') \\ & \wedge\ \text{UNCHANGED}\ wert \end{aligned}$$

Auch hier wird mittels *wertToBit* und *bitToVolt* zu dem von der Regelung bestimmten Stellwert (*wert*) die zugehörige Spannung (*volt*) und der entsprechende digitale Wert (*bitWert*) berechnet. Über den D/A-Wandler wird dieser an das Stellglied weitergeleitet. Der Stellwert bleibt dabei unverändert.

$$spec \triangleq \Box[putData]_{\langle wert, bitWert, volt\rangle}$$

Die Spezifikation besteht lediglich aus einer Sicherheitsbedingung, die besagt, daß nur mit *putData* die Zustandsvariablen für die Stellwerte verändert werden dürfen. Die entscheidende Aufgabe des Aktorprozesses in der Software besteht in der Überwachung, daß immer nur ein Prozeß Zugriff auf den Ausgangskanal hat. Hierzu gibt es in der Software die Befehle *lock* und *unlock*. Diese sind in der Spezifikation jedoch nicht enthalten, da durch die Aufteilung der entsprechenden Zustandsvariablen auf die verschiedenen Instanzen des Moduls *Aktorprozess* im Modul *Regelung* diese Sicherheitseigenschaft gewährleistet ist.

5.5 Modul Regelung

Siehe Anhang B.3 auf Seite 65

Die Regelung berechnet gemäß den aktuellen Meßwerten die neuen Stellwerte. Obwohl es für die Luftsättigung und den *pH* keine Stellglieder gibt, wird dennoch für beide ein Stellwert berechnet. Diese Stellwerte sind die Grundlage, auf der die einzelnen Gasstellwerte bestimmt werden. Die neuen Stellwerte werden anschließend über die Aktorprozesse an die D/A-Wandler weitergeleitet.

Zunächst wird für jeden Stellwert ein Aktorprozeß eingebunden.

import *Echtzeit*

include *Regler* **as** *regler* **with** $pH \leftarrow pHStell$
$pO_2 \leftarrow pO_2Stell$
include *Aktorprozess* **as** $O_2Aktor(Kanal_{O_2})$ **with** $wert \leftarrow O_2Stell$
include *Aktorprozess* **as** $N_2Aktor(Kanal_{N_2})$ **with** $wert \leftarrow N_2Stell$
include *Aktorprozess* **as** $CO_2Aktor(Kanal_{CO_2})$ **with** $wert \leftarrow CO_2Stell$
include *Aktorprozess* **as** $luft_1Aktor(Kanal_{luft_1})$ **with** $wert \leftarrow luft_1Stell$
include *Aktorprozess* **as** $pO_2Aktor(Kanal_{pO_2})$ **with** $wert \leftarrow pO_2Stell$
include *Aktorprozess* **as** $pHAktor(Kanal_{pH})$ **with** $wert \leftarrow pHStell$

Das Modul *Regler* liegt nicht als TLA$^+$ Spezifikation vor. Es dient der Berechnung der Stellwerte (pH und pO_2) gemäß der gewünschten Regelungsstrategie. Die Menge aller möglichen Strategien ist definiert als

$$Regeltypen \triangleq \{\text{“P”}, \text{“PD”}, \text{“PI”}, \text{“PID”}, \text{“FUZZY”}\}$$

Zur Durchführung der Stellwertneuberechnung werden die aktuellen Meßwerte benötigt. Daher wird in dem Modul *Regler* das Modul *Sensorprozess* eingebunden, um Zugriff auf die entsprechenden Zustandsvariablen zu haben.

Die einzige Action, die von diesem Modul zur Verfügung gestellt wird, ist *neuberechnung*. Diese beinhaltet auf Grund der besseren Übersichtlichkeit zwei weitere Actions, welche aber nicht eigenständig verwendet werden.

$$\begin{aligned} neuberechnung \triangleq\ & \wedge\ regler.berechnepHStell(pHRegler) \\ & \wedge\ regler.berechnepO_2Stell(pO_2Regler) \\ & \wedge\ mischung \\ & \wedge\ putData \\ & \wedge\ trigger' \neq trigger \end{aligned}$$

Die Neuberechnung aller Stellwerte besteht aus der Berechnung der Stellwerte pO_2Stell und $pHStell$ mittels der entsprechenden Action aus dem Modul *Regler* unter Berücksichtigung der Regelungsstrategie.

Die Action *mischung* sorgt für die Umsetzung dieser Stellwerte in die einzelnen Gasstellwerte.

Mittels *putData* werden diese Daten an die D/A-Wandler übergeben. Die Zustandsvariable *trigger* wird zum Takten der Neuberechnung verändert.

$$\begin{aligned} mischung \triangleq\ & \wedge\ CO_2Stell' = ((CO_2Max - CO_2Min) \cdot (1 - \tfrac{pHStell'}{14})) + CO_2Min \\ & \wedge\ pO_2Stell' = \tfrac{4.63 \cdot O_2Stell' + Luft_1Stell'}{VStromSoll} \\ & \wedge\ O_2Stell' = \textbf{if } pO_2Stell' \leq \tfrac{VStromSoll - CO_2Stell'}{VStromSoll} \cdot 21.6 \textbf{ then } 0 \\ & \wedge\ N_2Stell' = \textbf{if } O_2Stell' \neq 0 \textbf{ then } 0 \\ & \wedge\ VStromSoll = CO_2Stell' + O_2Stell' + N_2Stell' + luft_1Stell' \end{aligned}$$

Der CO_2 Anteil in der Membrangasmischung wird bestimmt durch den pH Stellwert. Hierbei verläuft der CO_2 Stellwert linear mit negativer Steigung zum pH Stellwert. Das bedeutet einen maximalen CO_2 Stellwert bei minimalem pH Stellwert. Steigt der pH Stellwert, so sinkt entsprechend der CO_2 Stellwert, bis sein Minimum erreicht ist.
Der verbleibende Volumenstromanteil wird für die Begasung mit Luft/N_2 oder Luft/O_2 verwendet. Der pO_2 Stellwert drückt den prozentualen Anteil von Luft am gesamten Volumenstrom aus. Der Luftvolumenstrom geht demnach mit dem Faktor 1, der Sauerstoffvolumenstrom mit dem Faktor 4.63[1] pro Volumeneinheit in den pO_2 Stellwert ein.
Für pO_2 Stellwerte, die durch reine Luftbegasung noch erreicht werden können, ist der Sauerstoffvolumenstrom 0.
Stickstoff soll nur verwendet werden, wenn kein Sauerstoff verwendet wird.
Insgesamt ergeben alle Gase zusammen den Gesamtvolumenstrom.

$$
\begin{array}{lll}
putData & \triangleq & \wedge\ O_2Aktor.putData \\
& & \wedge\ N_2Aktor.putData \\
& & \wedge\ CO_2Aktor.putData \\
& & \wedge\ luft_1Aktor.putData \\
& & \wedge\ pO_2Aktor.putData \\
& & \wedge\ pHAktor.putData
\end{array}
$$

Die neuen Stellwerte werden über die jeweiligen Aktorprozesse an die D/A-Wandler gegeben.

Die Spezifikation der Regelung sorgt mittels eines Timers x für eine zyklische Neuberechnung der Stellwerte innerhalb des Intervalls Δ. Der Zyklus der Meßwertbestimmung aus dem Modul *Sensorprozess* ist von diesem Zyklus unabhängig.

$$
\begin{array}{lll}
spec & \triangleq & \wedge\ \exists\, x\ :\ \wedge\ Vtimer(x, neuberechung, \Delta, trigger) \\
& & \qquad\ \wedge\ ExactTimer(x, neuberechnung, trigger) \\
& & \wedge\ \Box[neuberechnung]_{vars}
\end{array}
$$

Nur durch die Action *neuberechnung* ist eine Veränderung der Stellwerte möglich.

[1] Der Anteil von Sauerstoff in der Luft beträgt 21.6%, demnach beträgt die Luftsättigung von reinem Sauerstoff 463%

5.6 Modul SoftwareKontrolle

Siehe Anhang B.4 auf Seite 67

Die Kontrolle der Regelung besteht in der regelmäßigen Bestimmung der Meßwerte und der ebenso regelmäßigen Neuberechnung der Stellwerte. Diese beiden Prozesse, sind jedoch unabhängig voneinander realisiert. Zunächst wird für jeden Meßwert ein Sensorprozeß eingebunden. Hierbei werden alle Gasmeßwerte tatsächlich am Stellglied gemessen. Lediglich die beiden Werte Luftsättigung und pH besitzen kein Stellglied und werden daher über zwei spezielle Sonden ermittelt.

import *InitKonstanten*
import *Echtzeit*
include *Sensorprozess* **as** $O_2Sensor(Min_{O_2}, Max_{O_2}, \Delta_{O_2}, Kanal_{O_2}, Vers_{O_2}, Steig_{O_2})$
include *Sensorprozess* **as** $N_2Sensor(Min_{N_2}, Max_{N_2}, \Delta_{N_2}, Kanal_{N_2}, Vers_{N_2}, Steig_{N_2})$
include *Sensorprozess* **as** $CO_2Sensor(Min_{CO_2}, Max_{CO_2}, \Delta_{CO_2}, Kanal_{CO_2}, Vers_{CO_2}, Steig_{CO_2})$
include *Sensorprozess* **as** $luft_1Sensor(Min_{luft_1}, Max_{luft_1}, \Delta_{luft_1}, Kanal_{luft_1}, Vers_{luft_1}, Steig_{luft_1})$
include *Sensorprozess* **as** $pO_2Sensor(Min_{pO_2}, Max_{pO_2}, \Delta_{pO_2}, Kanal_{pO_2}, Vers_{pO_2}, Steig_{pO_2})$
include *Sensorprozess* **as** $pHSensor(Min_{pH}, Max_{pH}, \Delta_{pH}, Kanal_{pH}, Vers_{pH}, Steig_{pH})$
include *Regelung* **as** $regelung(VStrom, CO_2Max, \Delta_{StellNeu}, pO_2Regler, pHRegler,$
$Kanal_{O_2}, Kanal_{N_2}, Kanal_{CO_2}, Kanal_{luft_1}, Kanal_{pO_2}, Kanal_{pH})$

Das Modul *InitKonstanten* beinhaltet die Deklaration der Konstanten, die zur Initialisierung der Konstanten in den Modulen *Sensorprozess* und *Regelung* dienen.

$$
\begin{aligned}
spec \triangleq\ & \wedge\ O_2Sensor.spec \\
& \wedge\ N_2Sensor.spec \\
& \wedge\ CO_2Sensor.spec \\
& \wedge\ luft_1Sensor.spec \\
& \wedge\ pO_2Sensor.spec \\
& \wedge\ pHSensor.spec \\
& \wedge\ regelung.spec
\end{aligned}
$$

Durch die Spezifikation wird für jeden Stellwert ein Timer geschaffen, der die Filterung der Meßwerte taktet. Die Regelung sorgt dann mittels eines eigenen Timers für die regelmäßige Neuberechnung der Stellwerte.

Kapitel 6

Beurteilung der Regelungssoftware

Anhand der Spezifikation der Regelungssoftware und des Bioreaktors kann nun eine Beurteilung der Software erfolgen. Es handelt sich hierbei allerdings nicht um einen Verifikationsbeweis. Dieser wäre auf Grund der komplexen Problemstellung zu umfangreich, des weiteren fehlt für einen formalen Beweis eine vollständige Softwarebeschreibung. Dennoch ist es möglich, anhand der vorhandenen Spezifikationen eine Beweisführung zu skizzieren und dabei die Softwareimplementation zu beurteilen.

6.1 Anwendbarkeit der Spezifikationen

Vorab stellt sich die Frage, welche Teile der umfangreichen Reaktorspezifikation sich mit der Softwarespezifikation vergleichen lassen, da diese nur die Membrangasregelung und somit nur einen Ausschnitt der gesamten Kultivierung beschreibt. Es zeigt sich, daß eine solche Beurteilung erleichtert wird, wenn bei der Spezifikation von vage beschriebenen Systemen die hier dargestellte Aufteilung der Problemstellung vorgenommen wird.

Ziel des Prozesses. Welcher Endzustand bzw. welcher Verlauf des Prozeses soll mittels des Stoffes und der Kontrolle erreicht werden?
Die Beschreibung des Ziels erfolgt im Modul

- *Kultivierung*

Stoffe die zum Erreichen des Ziels verwendet werden.
Die Beschreibung der Stoffe erfolgt in dem Modul

- *Medium*

Kontrolle des Prozesses mittels Einwirkung auf den Stoff unter Berücksichtigung des Zieles.
Die Beschreibung der Kontrolle erfolgt in den Modulen

- *Bioreaktor*
- *Kontrolle*

Umwelt, die auf den Prozeß einwirkt, aber nicht über die Steuerung beeinflußbar ist.
Die Beschreibung der Umwelt erfolgt im Modul

- *Umgebung*

Die Softwareregelung der Gasmischung fällt in diesem Schema unter den Punkt Kontrolle. Daher sind zur Beurteilung der Software die unter dem Punkt Kontrolle eingeordneten Module des Bioreaktors zu betrachten. Man erkennt, daß die Software die vorhandenen Einwirkungsmöglichkeiten des Bioreaktors benutzt, daher ist ein Verlgeich mit dem Modul *Bioreaktor* nicht notwendig. Die Spezifikation der Software sollte also den Anforderungen genügen, die sich aus der Spezifikation des Moduls *Kontrolle* ergeben.

6.2 Refinement Mapping

Daß eine Spezifikation eine andere erfüllt, ist für TLA Spezifikationen gleichbedeutend mit einer Implikation. Mit dem Beweis des Ausdrucks

$$\exists\, ZV_{SW} \,:\, SoftwareKontrolle.spec \Rightarrow \exists\, ZV_{BR} \,:\, Kontrolle.spec$$

wird gezeigt, daß die Softwareregelung die Anforderungen erfüllt, die im Modul *Kontrolle* beschrieben sind.

Hierbei bezeichnet ZV_{BR} alle Zustandsvariablen aus dem Modul *Kontrolle*. Entsprechendes gilt für ZV_{SW} bei der Softwarespezifikation. Um diese Implikation beweisen zu können, ist es erforderlich, die Zustandsvariablen der Reaktorspezifikation auf die Zustandsvariablen der Softwarespezifikation abzubilden. Diese mit dem Namen *refinement mapping* bezeichnete Zuweisung ist in dem vorliegenden Fall ein nicht triviales Problem. Die Reaktorspezifikation enthält nur Zustandsvariablen für die momentanen Istwerte. Die Softwarebeschreibung basiert jedoch auf der Bestimmung von entsprechenden Stellwerten. Diese dienen als Richtwert für die zukünftigen Istwerte der Gasmischung. Unter der Annahme einer korrekten Umsetzung der Stellwerte sollte ein *refinement mapping* für die Gasmischung durch eine direkte Abbildung des Istwertes auf den Stellwert möglich sein. Die Stellwerte für pH

und pO_2 besitzen, da es sich hierbei um abstrakte Stellwerte handelt, jedoch keinen Gegenpart in der Reaktorbeschreibung. Für einen Beweis könnte daher ein *refinement mapping* für diese Zustandsvariablen unnötig sein. Auch für die in der Software enthaltenen Werte für die Reglerspannung und die digitale Repräsentation der Stellwerte kommt ein *refinement mapping* nicht in Betracht. Entsprechendes gilt bei der Reaktorspezifikation für alle nicht von der Softwareregelung betroffenen Zustandsvariablen. Möglicherweise ist es empfehlenswert, für die Regelung des Bioreaktors als auch für die Software weitere Spezifikationen zu schreiben, in denen die entsprechenden Zustandsvariablen und Definitionen aus den Abschnitten *actions* und *temporal* nicht enthalten sind. Diese verkleinerten Module müßten zwar auch mittels eines entsprechenden Beweises verifiziert werden, so daß in diesem Beweis ebenso mit dem Problem des Weglassens von Zustandsvariablen umgegangen werden müßte. Da sich das Problem aber auf zwei kleinere Teilprobleme verteilt und es sich hierbei um eine Abstraktion und nicht um einen Vergleich zweier völlig verschiedener Beschreibungen handelt, könnte möglicherweise eine Vereinfachung des *refinement mapping* erreicht werden.

6.3 Sicherheitseigenschaften

Betrachtet man die Spezifikationen genauer, so läßt sich feststellen, daß sie dem Aufbau

$$\begin{aligned} sepc \;\triangleq\; & \wedge\; Init \\ & \wedge\; \Box[actions]_v \end{aligned}$$

entsprechen.

Dies ist die kanonische Form von TLA$^+$ Spezifikationen. In den Spezifikationen der meisten Bioreaktormodule sind noch zusätzliche Ausdrücke für Timer-, Echtzeit- und Kontinuitätsbedingungen enthalten. Jedoch ist in [Lam93] gezeigt worden, daß sich die Bedingungen für Timer und Echtzeit in eine kanonische Form überführen lassen. Ob auch für die Kontinuitätsaussage eine solche Überführung möglich ist, müßte gezeigt werden. Ist dies möglich, so kann man die Spezifikationen für die Reaktorkontrolle und die Softwarekontrolle in eine kanonische Form umwandeln.

Die Implikation läßt sich dann beschreiben durch

$$(\exists\, ZV_{SW} \;:\; Init_{SW} \wedge \Box[actions_{SW}]_{ZV_{SW}}) \Rightarrow (\exists\, ZV_{BR} \;:\; Init_{BR} \wedge \Box[actions_{BR}]_{ZV_{BR}})$$

Der Beweis dieses Ausdrucks reduziert sich nach dem Beweisschema, wie es in [Lad95] beschrieben ist, auf den Beweis von

1. $Init_{SW} \Rightarrow \overline{Init_{BR}}$
2. $Init_{SW} \Rightarrow Inv$
3. $Inv \wedge [actions_{SW}] \Rightarrow Inv'$
4. $[actions_{SW} \wedge Inv \wedge Inv']_{ZV_{SW}} \Rightarrow [\overline{actions_{BR}}]_{\overline{ZV_{BR}}}$
5. $\Box Inv \wedge \Box[actions_{SW}]_{ZV_{SW}} \wedge L_{SW} \Rightarrow \overline{L_{BR}}$

Die mit einem Überstrich versehenen Bezeichner stehen für die ursprünglichen Ausdrücke, in denen jedoch alle Zustandsvariablen durch das entsprechende *refinement mapping* ersetzt wurden.

Der Beweis zu 1. sollte bei gleicher Anfangsbelegung der Zustandsvariablen direkt möglich sein.

Der Punkt 2. besagt, daß aus dem Initialzustand eine Invariante folgt, die als Bindeglied über den gesamten Beweis unverändert bleibt, also in jedem Zustand wahr ist. Das Bestimmen einer solchen Invariante ist eine nicht triviale Angelegenheit und erfordert eine entsprechende Erfahrung des Beweisführenden. Es ist daher schwer abzuschätzen, wie diese lautet, ohne den Beweis wirklich durchzuführen. Eine Mindestanforderung an diese Invariante ist möglicherweise:

$\Box(VStromSoll = CO_2Stell + O_2Stell + N_2Stell + luft_1Stell)$

Hat man die richtige Invariante gefunden, so läßt sich 3. meist leicht folgern.

Durch den Beweis von 4., wird gezeigt, daß es zu jeder Action aus $actions_{SW}$ einen entsprechenden Gegenpart in $\overline{actions_{BR}}$ gibt. Betrachtet man im Modul *Kontrolle* nur noch die Actions, die an der Membrangaszusammensetzung beteiligt sind, $pH\_korrektur$ und $pO_2\_korrektur$, zusammen mit den entsprechenden Timerbedingungen, $pH\_richtig$ und $pO_2\_richtig$, so sollte in der Beschreibung der Regelungssoftware eine korrespondierende Action vorhanden sein. Da die Action *neuberechnung* aus dem Modul *Regelung* die Bestimmung der Membrangaszusammensetzung behandelt, wird eine Überprüfung der *neuberechnung* auf ihre Integrität mit den o.g. Actions aus dem Moduls *Kontrolle* erforderlich sein.

Ein solcher informeller Vergleich zeigt, daß die erste Forderung aus der Reaktorkontrollspezifikation, die Summe aller Gase entspricht dem Gesamtvolumenstrom, durch die Software offensichtlich erfüllt wird, da dies auch in der *neuberechnung* gefordert wird.

Die zweite Forderung ist das Erreichen des Sollwertes. Die Erfüllung dieser Forderung hängt von dem Regelalgorithmus ab, welcher den PID- bzw. Fuzzy-Regler implementiert. Die Spezifikation dieses Reglers ist hier durch das nicht näher beschriebene Modul *Regler* dargestellt, welches in *Regelung* eingebunden wird. Da dieses Modul nicht existiert, läßt sich nur die allgemeingültige Aussage treffen, daß das Erreichen des Sollwertes abhängig ist von dem berechneten Stellwert und dieser wiederum von der korrekten Implementation eines Reglermodells inklusive der richtigen Auswahl der Steuerungsparameter. Diese sind im Falle des PID-Reglers die Reglerparameter K_P, T_N und T_V, im Falle eines Fuzzy-Reglers sind dies die Regelbasis, die Fuzzyfizierungs- und Defuzzifizierungsmethode. Eine formale Verifizierung wird hierbei durch die Tatsache erschwert, daß die Einstellung der Reglerparameter nicht ein integrierter Bestandteil der Software ist, sondern vom Benutzer über eine Initialisierungsdatei frei gewählt werden kann. Gleiches gilt für die Regelbasis der Fuzzy-Regelung.

Ein Vergleich der Timerbedingungen aus beiden Spezifikationen zeigt, daß der maximale Zeitraum, über den der Istwert vom Sollwert abweichen darf – im Modul *Kontrolle* mit Δ_{pH} und Δ_{pO_2} bezeichnet – das im Modul *Regelung* benutzte Δ beherscht. Dieses Δ bestimmt die Zeitintervalle zwischen den Neuberechnungen der Stellwerte. Ebenso verhält es sich mit dem Δ für die Wiederholung der Meßwertbestimmung aus dem Modul *Sensorprozess*. Da diese Zeitintervalle, wie bei den Reglerparametern, vom Benutzer frei gewählt werden können, ist eine Beweismöglichkeit für die Erfüllung der Reaktorkontrollspezifikation fraglich. Hierzu müßte die Software um eine bisher nicht vorhandene Überprüfung der gewählten Benutzereinstellungen erweitert werden.

In den Spezifkationen gibt es keine Lebendigkeitsaussagen L_{SW} und $\overline{L_{BR}}$. Nimmt man für beide den Wert TRUE an, so erübrigt sich der Beweis von 5., da die Aussage $\Box Inv$ für eine Invariante TRUE sein sollte und der Ausdruck $\Box[actions_{SW}]_{ZV_{SW}}$ laut Spezifikation gilt.

Zusätzlich zu den Punkten aus dem Beweisschema wird in der Spezifikation der Kontrolle gefordert, daß die Luftsättigung, der pH, die Mediumtemperatur und die Rührerdrehzahl bestimmte untere und obere Schraken nicht verletzen, da sonst die Möglichkeit des Absterbens der Zellkultur be-

steht. Entsprechend der Reglerparameter werden diese Grenzwerte bei der Softwaresteuerung in einer Initialisierungsdatei vom Benutzer festgelegt und unterliegen daher keiner Kontrolle der Software.

6.4 Ergebnis

Um einen formalen Beweis durchführen zu können, ist es notwendig, eine vollständige Spezifikation der Software zu erstellen. Zusätzlich könnte eine Abstraktion der vorhandenen Spezifikationen sowohl für die Reaktorkontrolle als auch für die Softwareregelung der Membrangasmischung bei der Durchführung des *refinement mappings* hilfreich sein. Dies gilt insbesondere dann, wenn diese abstrakteren Spezifikationen nur noch die Gasregelung betreffende Zustandsvariablen enthalten.

Um das o.g. Beweisschema anwenden zu können, müßte überprüft werden, ob sich die Beschreibung des kontinuierlichen Verhaltens von Zustandsvariablen in der kanonischen Form darstellen läßt.

Mittels der dargestellten Beweisskizze und der daraus resultierenden Analyse der entsprechenden Module ist es jedoch auch ohne formale Beweisführung möglich, die Software auf die Erfüllung der Reaktorspezifikation zu überprüfen. Es zeigt sich, daß die Implementation den Anforderungen der Kultivierungssteuerung genügt, sofern diese der Kontrolle der Software unterliegen. Es hat sich jedoch auch gezeigt, daß auf Grund einer fehlenden Softwarekontrolle über die beliebige Auswahl der Regelungsparameter, ein korrektes Arbeiten der Software nicht bewiesen werden kann. Durch eine ungünstige Wahl der Stellwertgrenzen, zusammen mit einer ebenso ungünstigen Auswahl an Regelungsparametern ist sogar eine Zerstörung der Zellkultur möglich. Um diese Sicherheitslücke zu schließen, müßte die vorhandene Software entsprechend erweitert werden.

Die hier aufgezeigten Sicherheitslücken führen zwar nur bei einem sehr ungünstigen Zusammenspiel verschiedener Faktoren zu einem fehlerhaften Funktionieren der Softwareregelung, doch gerade wegen des seltenen Auftretens solcher Situationen ist es zur Vermeidung dieser Fehlersituationen wichtig, die genannen Sicherheitslücken der Software zu kennen.

Kapitel 7

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wird ein in der Praxis genutztes hybrides System, welches nur vage[1] beschrieben ist, mittels einer formalen Beschreibungstechnik spezifiziert. In diesem Zusammenhang wird die Eignung der Spezifikationssprache TLA$^+$ für eine solche Spezifikation am Beispiel eines 20 Liter Perfusionsbioreaktors gezeigt. Anhand dieser TLA$^+$ Spezifikation und der TLA$^+$ Spezifikation einer existierenden Regelungssoftware für die Begasung des Reaktors wird mittels einer Skizze des Implementationsbeweises gezeigt, welche Sicherheitslücken die Implementation aufweist.

Trotz der nur vagen Beschreibung der unterschiedlichen Komponenten des Bioreaktors ist es gelungen, eine inhaltlich zutreffende und formal korrekte[2] TLA$^+$ Spezifikation zu schreiben. Dabei konnte auf dem Abstraktionsniveau des den Reaktor benutzenden Biologen verblieben werden. Alle technischen Komponenten werden nur so detailliert beschrieben, wie es die Benutzung des Bioreaktors zur Kultivierung erfordert. Das hierzu benötigte biologische Fachwissen wurde in mehreren Interviews mit einem wissenschaftlichen Mitarbeiter der AG Zellkulturtechnik ermittelt.

Auf Grund des hohen Abstraktionsniveaus ist es auch möglich, der aktuellen philosophischen Grundlagenforschung zum Thema formale Beschreibungen gerecht zu werden, welche unter anderem von Searle [Sea96] und Ladkin [Lad97.1] in der Behandlung von *artifacts* betrieben wird. Dies spiegelt sich in der Einteilung des Reaktorsystems in die einzelnen Module wider. In diesem Zusammenhang ist es sowohl bei der Erstellung

[1] Zur Bedeutung des Wortes *vage* siehe Abschnitt 1.2 auf Seite 2

[2] „formal korrekt“ bedeutet hier, daß ausschliesslich TLA$^+$ Sprachelemente verwendet wurden und somit eine klare Zuordnung von Syntax und Semantik für die gesamte Spezifikation sichergestellt ist.

von Spezifikationen zu vagen Systemen als auch bei der anschliessenden Beweisskizzierung vorteilhaft, eine Unterteilung in die Punkte Ziel, Stoffe, Kontrolle und Umwelt vorzunehmen.

Im Zusammenhang mit der Spezifikation des Reaktors, mußten die den Kultivierungsprozeß bestimmenden Echtzeitbedingungen berücksichtigt werden. Hierzu wurde die bewährte Verfahrensweise zur Beschreibung von Echtzeit in TLA verwendet und um den Aspekt des kontinuierlichen Verlaufes von Zustandsvariablen erweitert. Indem mit der Reaktorspezifikation eine existierende Softwaresteuerung auf ihre Sicherheitseigenschaften überprüft wurde, wird der Nutzen von formalen Beschreibungen für vage Problemstellungen belegt. Diese Prüfung erfolgt durch die Skizzierung des Beweises, daß die TLA$^+$ Spezifikation der Software die Spezifikation des Reaktors impliziert und somit implementiert. Hierbei zeigt sich, daß in der Softwaresteuerung bisher nicht berücksichtigte Sicherheitslücken existieren, die in ungünstigen Fällen zur Zerstörung der Zellkultur führen könnten.

Die erzielten Ergebnisse zeigen, daß sich TLA$^+$ neben den bisherigen Einsatzmöglichkeiten auch zur formalen Beschreibung von vagen Problemen eignet und benutzt werden sollte, da sich auf diese Weise die Sicherheit von Softwareimplementationen steigern läßt.

Aufbauend auf den hier erzielten Ergebnissen, wäre eine vollständige Spezifikation der Software wünschenswert, um mittels eines durchgeführten Implementationsbeweises mögliche weitere Sicherheitslücken der Software zu ermitteln oder aber die Fehlerfreiheit der Software zu beweisen. Vorab müßte jedoch gezeigt werden, daß die hier eingeführte Behandlung von kontinuierlichen Zustandsvariablen in der kanonischen Form für TLA Spezifikationen darstellbar ist. Die intensive Weiterentwicklung von automatischen Beweisern und „proofcheckers“ zur Vereinfachung oder Automatisierung von Implementationsbeweisen sowie die Schaffung von Werkzeugen zur Erstellung von TLA$^+$ Spezifikationen sind weitere Aufgaben für die zukünftige Forschung.

Anhang A

Die Module zum Bioreaktor

A.1 Modul MiscDef

module *MiscDef*

import *Reals*
import *Naturals*
import *Sequences*

definitions

$Real^{+} \triangleq \{x \in Real : x > 0\}$

$Real^{0+} \triangleq \{x \in Real : x \geq 0\}$

$Nat^{+} \triangleq \{n \in Nat : n > 0\}$

$Nat^{0+} \triangleq \{n \in Nat : n \geq 0\}$

$|z| \triangleq$ **if** $z < 0$ **then** $-z$ **else** z

$min(a, b) \triangleq$ **if** $a \leq b$ **then** a **else** b

$max(a, b) \triangleq$ **if** $a \geq b$ **then** a **else** b

A.2 Modul Echtzeit

module *Echtzeit*

parameters

now : VARIABLE

∞ : CONSTANT

import *MiscDef*

assumption

$InfinityUnReal \triangleq \infty \notin Real$

actions

$$RT(v) \triangleq \land\ now \in Real \land \Box[\land\ now' \in \{r \in Real : r > now\} \land\ v' = v]_{now} \land \forall\, t \in Real : \Diamond(now > t)$$

$$VTimer(x, A, \delta, v) \triangleq \land\ x = \textbf{if } \textsc{Enabled}\ \langle A \rangle_v \textbf{ then } now + \delta \textbf{ else } \infty \land \Box[x' = \textbf{if } (\ \textsc{Enabled}\ \langle A \rangle_v)' \textbf{ then if } \langle A \rangle_v \textbf{ then } now' + \delta \textbf{ else } x \textbf{ else } \infty\ ]_{\langle x, v \rangle}$$

$$MaxTimer(x) \triangleq \Box[(x \neq \infty) \Rightarrow (now' \leq x)]_{now}$$

$$ExactTimer(x, A, v) \triangleq \Box[\langle A \rangle_v \Rightarrow (now = x)]_v$$

$$kontinuierlich(v) \triangleq \forall\, i \in 1..Len(v) : \Box[\forall\, \delta \in Real^+ : \exists\, \epsilon \in Real^+ : now' - now < \epsilon \Rightarrow |v[i]' - v[i]| < \delta]_{v[i]}$$

A.3 Modul Umgebung

module *Umgebung*

parameters

$temperatur, helligkeit$: VARIABLE
$MaxHelligkeit$: CONSTANT

definitions

$tempBed \triangleq 10°C \leq temperatur \leq 35°C$
$lichtBed \triangleq helligkeit \leq MaxHelligkeit$

temporal

$vars \triangleq \langle temperatur, helligkeit \rangle$
$spec \triangleq \wedge \Box(tempBed \wedge lichtBed)$
$\wedge\ RT(vars)$
$\wedge\ kontinuierlich(vars)$

A.4 Modul Medium

module *Medium*

parameters

$aktiveZellen, toteZellen, bakterien : Nat^{0+}$ VARIABLE
$pH, pO_2, temperatur, raumtemperatur : Real^{+}$ VARIABLE
$medium : Zutaten \cup Gifte \to Real^{0+}$ VARIABLE
$Zutaten, Gifte, Medium_0, pH_0, pO_{2(0)}, temperatur_0$: CONSTANT
$AktiveZellen_0, ToteZellen_0$: CONSTANT

---

import *Echtzeit*
include *Umgebung* **as** *umwelt* **with** $raumtemperatur \leftarrow temperatur$

---

predicates

$$
\begin{aligned}
Init \triangleq\ & \wedge\ aktiveZellen = AktiveZellen_0 \\
& \wedge\ toteZellen = ToteZellen_0 \\
& \wedge\ medium = Medium_0 \\
& \wedge\ pH = pH_0 \\
& \wedge\ pO_2 = pO_{2(0)} \\
& \wedge\ temperatur = temperatur_0 \\
& \wedge\ bakterien = 0
\end{aligned}
$$

---

actions

$$
\begin{aligned}
wachstum \triangleq\ & \wedge\ aktiveZellen' = aktiveZellen + 1 \\
& \wedge\ \text{UNCHANGED}\ \langle raumtemperatur, toteZellen, medium, pH, pO_2, \\
& \qquad temperatur, bakterien \rangle
\end{aligned}
$$

$$
\begin{aligned}
sterben \triangleq\ & \wedge\ aktiveZellen > 0 \\
& \wedge\ aktiveZellen' = aktiveZellen - 1 \\
& \wedge\ toteZellen' = toteZellen + 1 \\
& \wedge\ \text{UNCHANGED}\ \langle raumtemperatur, medium, pH, pO_2, temperatur, bakterien \rangle
\end{aligned}
$$

$$
\begin{aligned}
verbrauch \triangleq\ & \wedge\ \exists\, z \in Zutaten : 0 \leq medium'[z] < medium[z] \\
& \wedge\ \forall\, x \in Zutaten \cup Gifte : (x = z \vee \text{UNCHANGED}\ medium[x]) \\
& \wedge\ \text{UNCHANGED}\ \langle raumtemperatur, aktiveZellen, toteZellen, \\
& \qquad pH, pO_2, temperatur, bakterien \rangle
\end{aligned}
$$

$$
\begin{aligned}
ausscheidung \triangleq\ & \wedge\ \exists\, g \in Gifte : medium'[g] > medium[g] \\
& \wedge\ \forall\, x \in Zutaten \cup Gifte : (x = g \vee \text{UNCHANGED}\ medium[x]) \\
& \wedge\ \text{UNCHANGED}\ \langle raumtemperatur, aktiveZellen, toteZellen, \\
& \qquad pH, pO_2, temperatur, bakterien \rangle
\end{aligned}
$$

$$
\begin{array}{lcl}
pHaenderung & \triangleq & \land\ 0 \leq pH' \leq 14 \\
 & & \land\ pH' \neq pH \\
 & & \land\ \text{UNCHANGED}\ \langle raumtemperatur, aktiveZellen, toteZellen, medium, \\
 & & \qquad pO_2, temperatur, bakterien \rangle \\
pO_2aenderung & \triangleq & \land\ 0 \leq pO_2' \leq 460 \\
 & & \land\ pO_2' \neq pO_2 \\
 & & \land\ \text{UNCHANGED}\ \langle raumtemperatur, aktiveZellen, toteZellen, medium, \\
 & & \qquad pH, temperatur, bakterien \rangle \\
abkuehlung & \triangleq & \land\ raumtemperatur \leq temperatur' < temperatur \\
 & & \land\ \text{UNCHANGED}\ \langle raumtemperatur, aktiveZellen, toteZellen, \\
 & & \qquad medium, pH, pO_2, bakterien \rangle \\
baktWachstum & \triangleq & \land\ bakterien > 0 \\
 & & \land\ bakterien' > bakterien \\
 & & \land\ \text{UNCHANGED}\ \langle raumtemperatur, aktiveZellen, toteZellen, \\
 & & \qquad medium, pH, pO_2, temperatur \rangle
\end{array}
$$

---

temporal

$$
\begin{array}{lcl}
vars1 & \triangleq & \langle pH, pO_2, temperatur, raumtemperatur \rangle \\
vars & \triangleq & vars1 \circ \langle medium, aktiveZellen, toteZellen, bakterien \rangle \\
actions & \triangleq & \lor\ wachstum \\
 & & \lor\ sterben \\
 & & \lor\ verbrauch \\
 & & \lor\ pHaenderung \\
 & & \lor\ pO_2aenderung \\
 & & \lor\ abkuehlung \\
 & & \lor\ baktWachstum \\
spec & \triangleq & \land\ Init \\
 & & \land\ \Box[actions]_{vars} \\
 & & \land\ RT(vars) \\
 & & \land\ kontinuierlich(vars1)
\end{array}
$$

A.5 Modul Bioreaktor

module *Bioreaktor*

parameters

$ruehrerdrehzahl, membrandruck, membrangas, membrangasfluss : Real^{0+}$ VARIABLE
$temperatur, kopfraumdruck, kopfraumgas, kopfraumgasfluss, volumen : Real^{0+}$ VARIABLE
$aktiveZellen, toteZellen, bakterien : Nat^{0+}$ VARIABLE
$medium : Zutaten \to Real^{0+}$ VARIABLE
$Zutaten, Volumen_0, Drehzahl_0$: CONSTANT
$Membrandruck_0, Membrangas_0, Membrangasfluss_0$: CONSTANT
$Kopfraumdruck_0, Kopfraumgas_0, Kopfraumgasfluss_0$: CONSTANT

---

import *Echtzeit*
include *Medium* **as** *medium*

---

definitions

$Maxvolumen \triangleq 20$

---

predicates

$$\begin{aligned} Init \triangleq\ & \wedge\ ruehrerdrehzahl = Drehzahl_0 \\ & \wedge\ volumen = Volumen_0 \\ & \wedge\ membrandruck = Membrandruck_0 \\ & \wedge\ membrangas = Membrangas_0 \\ & \wedge\ membrangasfluss = Membrangasfluss_0 \\ & \wedge\ kopfraumdruck = Kopfraumdruck_0 \\ & \wedge\ kopfraumgas = Kopfraumgas_0 \\ & \wedge\ kopfraumgasfluss = Kopfraumgasfluss_0 \end{aligned}$$

---

actions

$$\begin{aligned} neueZellen(a,t) \triangleq\ & \wedge\ a \in Nat^{+} \\ & \wedge\ t \in Nat^{0+} \\ & \wedge\ aktiveZellen' = aktiveZellen + a \\ & \wedge\ toteZellen' = toteZellen + t \\ & \wedge\ volumen < volumen' \leq Maxvolumen \\ & \wedge\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\ & \qquad membrangas, membrangasfluss, kopfraumgas, \\ & \qquad kopfraumgasfluss, medium, temperatur, bakterien \rangle \end{aligned}$$

$$
\begin{array}{lcl}
neuesMedium(m) & \triangleq & \land\ m \in Zutaten \to Real^{0+} \\
& & \land\ \forall\, x \in Zutaten\ :\ medium[x]' = medium[x] + m[x] \\
& & \land\ volumen < volumen' \leq Maxvolumen \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangas, membrangasfluss, kopfraumgas, \\
& & \qquad kopfraumgasfluss, aktiveZellen, toteZellen, \\
& & \qquad temperatur, bakterien \rangle \\
entnahme(a,t,m) & \triangleq & \land\ t, a \in Nat^{0+} \\
& & \land\ a \leq aktiveZellen \\
& & \land\ t \leq toteZellen \\
& & \land\ m \in Zutaten \to Real^{+} \\
& & \land\ aktiveZellen' = aktiveZellen - a \\
& & \land\ toteZellen' = toteZellen - t \\
& & \land\ \forall\, x \in Zutaten\ :\ \land\ medium[x]' = medium[x] - m[x] \\
& & \qquad\qquad\qquad\ \land\ 0 \leq medium[x]' \\
& & \land\ 0 \leq volumen' < volumen \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangas, membrangasfluss, kopfraumgas, \\
& & \qquad kopfraumgasfluss, temperatur, bakterien \rangle \\
verseuchung(n) & \triangleq & \land\ n \in Nat^{+} \\
& & \land\ bakterien' = bakterien + n \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangas, membrangasfluss, kopfraumgas, \\
& & \qquad kopfraumgasfluss, aktiveZellen, toteZellen, \\
& & \qquad medium, temperatur, volumen \rangle \\
heizen(t) & \triangleq & \land\ t > temperatur \\
& & \land\ temperatur' = t \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangas, membrangasfluss, kopfraumgas, \\
& & \qquad kopfraumgasfluss, aktiveZellen, toteZellen, \\
& & \qquad medium, bakterien, volumen \rangle \\
drehzahl(d) & \triangleq & \land\ d \in Real^{0+} \\
& & \land\ d \neq ruehrerdrehzahl \\
& & \land\ ruehrerdrehzahl' = d \\
& & \land\ \text{UNCHANGED}\ \langle kopfraumdruck, membrandruck, membrangas, \\
& & \qquad membrangasfluss, kopfraumgas, kopfraumgasfluss, \\
& & \qquad aktiveZellen, toteZellen, medium, \\
& & \qquad temperatur, bakterien, volumen \rangle
\end{array}
$$

$$
\begin{array}{lcl}
mGas(co, o, n, l) & \triangleq & \land\ co, o, n, l \in Real^{0+} \\
& & \land\ co + o + n + l = 100 \\
& & \land\ \langle co, o, n, l \rangle \neq membrangas \\
& & \land\ membrangas' = \langle co, o, n, l \rangle \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangasfluss, kopfraumgasfluss, aktiveZellen, \\
& & \qquad toteZellen, medium, temperatur, \\
& & \qquad bakterien, volumen, kopfraumgas \rangle \\
kGas(wahl) & \triangleq & \land\ wahl \in \{\text{“luft”}, \text{“gemisch”}\} \\
& & \land\ kopfraumgas' = \textbf{if}\ wahl\ = \text{“luft”}\ \textbf{then}\ \langle 0, 0, 0, 100 \rangle \\
& & \qquad\qquad\qquad\qquad\qquad\qquad \textbf{else}\ \ membrangas' \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangasfluss, kopfraumgasfluss, aktiveZellen, \\
& & \qquad toteZellen, medium, temperatur, \\
& & \qquad bakterien, volumen, membrangas \rangle \\
mFluss(m) & \triangleq & \land\ m \in Real^{0+} \\
& & \land\ m \neq membrangasfluss \\
& & \land\ membrangasfluss' = m \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangas, kopfraumgas, aktiveZellen, toteZellen, \\
& & \qquad medium, temperatur, bakterien, volumen, \\
& & \qquad kopfraumgasfluss \rangle \\
kFluss(m) & \triangleq & \land\ m \in Real^{0+} \\
& & \land\ m \neq kopfraumgasfluss \\
& & \land\ kopfraumgasfluss' = m \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, \\
& & \qquad membrangas, kopfraumgas, aktiveZellen, toteZellen, \\
& & \qquad medium, temperatur, bakterien, volumen, \\
& & \qquad membrangasfluss \rangle \\
mDruck(d) & \triangleq & \land\ d \in Real^{+} \\
& & \land\ d \neq membrandruck \\
& & \land\ membrandruck' = d \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, membrangas, membrangasfluss, \\
& & \qquad kopfraumgas, kopfraumgasfluss, kopfraumdruck, \\
& & \qquad aktiveZellen, toteZellen, medium, \\
& & \qquad temperatur, bakterien, volumen \rangle \\
kDruck(d) & \triangleq & \land\ d \in Real^{+} \\
& & \land\ d \neq kopfraumdruck \\
& & \land\ kopfraumdruck' = d \\
& & \land\ \text{UNCHANGED}\ \langle ruehrerdrehzahl, membrangas, membrangasfluss, \\
& & \qquad kopfraumgas, kopfraumgasfluss, membrandruck \\
& & \qquad aktiveZellen, toteZellen, medium, \\
& & \qquad temperatur, bakterien, volumen \rangle
\end{array}
$$

temporal

$$
\begin{array}{lcl}
vars1 & \triangleq & \langle ruehrerdrehzahl, kopfraumdruck, membrandruck, membrangas, membrangasfluss, \\
 & & \quad kopfraumgas, kopfraumgasfluss, temperatur, volumen \rangle \\
vars & \triangleq & vars1 \circ \langle medium, aktiveZellen, toteZellen, bakterien \rangle \\
actions & \triangleq & \lor \exists\, a, t : neueZellen(a, t) \\
 & & \lor \exists\, m : neuesMedium(m) \\
 & & \lor \exists\, a, t, m : entnahme(a, t, m) \\
 & & \lor \exists\, n : verseuchung(n) \\
spec & \triangleq & \land\ Init \\
 & & \land\ \Box[actions]_{vars} \\
 & & \land\ RT(vars) \\
 & & \land\ kontinuierlich(vars1)
\end{array}
$$

A.6 Modul Kontrolle

module *Kontrolle*

parameters

$pH, pO_2, temperatur, ruehrerdrehzahl$: $Real^{0+}$ VARIABLE
$membrandruck, membrangasfluss$: $Real^{0+}$ VARIABLE
$kopfraumdruck, kopfraumgasfluss$: $Real^{0+}$ VARIABLE
$pH^s, pO_2^s, temp^s, drehz^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s$: VARIABLE
$Min\_pO_2, Max\_pO_2, Min\_pH, Max\_pH, Min\_temp, Max\_temp$: CONSTANT
Min_drehz, Max_drehz : CONSTANT
$\Delta_{pH}, \Delta_{pO_2}, \Delta_{temp}, \Delta_{drehz}, \Delta_{mDruck}, \Delta_{kDruck}, \Delta_{mFluss}, \Delta_{kFluss}$: CONSTANT
$pH_0^s, {pO_2}_{(0)}^s, temp_0^s, drehz_0^s, mDruck_0^s, mFluss_0^s, kDruck_0^s, kFluss_0^s$: CONSTANT

definitions

$sollVars \triangleq \langle pH^s, pO_2^s, temp^s, drehz^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s \rangle$

import *Echtzeit*
include *Bioreaktor* **as** *reaktor*

predicates

$$
\begin{aligned}
Init \triangleq\ & \wedge\ pH^s = pH_0^s \\
& \wedge\ pO_2^s = {pO_2}_{(0)}^s \\
& \wedge\ temp^s = temp_0^s \\
& \wedge\ drehz^s = drehz_0^s \\
& \wedge\ mDruck^s = mDruck_0^s \\
& \wedge\ mFluss^s = mFluss_0^s \\
& \wedge\ kDruck^s = kDruck_0^s \\
& \wedge\ kFluss^s = kFluss_0^s
\end{aligned}
$$

actions

$$
\begin{aligned}
pH\_soll(s) \triangleq\ & \wedge\ 0 \leq s \leq 14 \\
& \wedge\ pH^{s\prime} = s \\
& \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& \qquad pO_2^s, temp^s, drehz^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s \rangle
\end{aligned}
$$

$$
\begin{array}{lcl}
pO_2\_soll(s) & \triangleq & \wedge\ 0 \leq s \leq 460 \\
& & \wedge\ pO_2^{s\prime} = s \\
& & \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, temp^s, drehz^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s \rangle \\
temp\_soll(s) & \triangleq & \wedge\ 10 \leq s \leq 80 \\
& & \wedge\ temp^{s\prime} = s \\
& & \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, drehz^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s \rangle \\
drehz\_soll(s) & \triangleq & \wedge\ 0 \leq s \leq 50 \\
& & \wedge\ drehz^{s\prime} = s \\
& & \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, temp^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s \rangle \\
mDruck\_soll(s) & \triangleq & \wedge\ 0 \leq s \\
& & \wedge\ mDruck^{s\prime} = s \\
& & \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, temp^s, drehz^s, mFluss^s, kDruck^s, kFluss^s \rangle \\
kDruck\_soll(s) & \triangleq & \wedge\ 0 \leq s \\
& & \wedge\ kDruckd^{s\prime} = s \\
& & \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, temp^s, drehz^s, mDruck^s, mFluss^s, kFluss^s \rangle \\
mFluss\_soll(s) & \triangleq & \wedge\ 0 \leq s \\
& & \wedge\ mFluss^{s\prime} = s \\
& & \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, temp^s, drehz^s, mDruck^s, kDruck^s, kFluss^s \rangle \\
kFluss\_soll(s) & \triangleq & \wedge\ 0 \leq s \\
& & \wedge\ kFluss^{s\prime} = s \\
& & \wedge\ \text{UNCHANGED}\ \langle pH, pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, temp^s, drehz^s, mDruck^s, mFluss^s, kDruck^s \rangle \\
pH\_korrektur & \triangleq & \wedge\ pH \neq pH^s \\
& & \wedge\ \exists\, co, o, n, l\ :\ reaktor.mGas(co, o, n, l) \\
& & \wedge\ \exists\, w\ :\ reaktor.kGas(w) \\
& & \wedge\ pH' = pH^s \\
& & \wedge\ \text{UNCHANGED}\ \langle pO_2, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, temp^s, drehz^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s \rangle
\end{array}
$$

$$
\begin{array}{lll}
pO_2\_korrektur & \triangleq & \land\ pO_2 \neq pO_2^s \\
& & \land\ \exists\, co, o, n, l : reaktor.mGas(co, o, n, l) \\
& & \land\ \exists\, w : reaktor.kGas(w) \\
& & \land\ pO_2' = pO_2^s \\
& & \land\ \text{UNCHANGED}\ \langle pH, temperatur, ruehrerdrehzahl, membrandruck, \\
& & \qquad membrangasfluss, kopfraumdruck, kopfraumgasfluss, \\
& & \qquad pH^s, pO_2^s, temp^s, drehz^s, mDruck^s, mFluss^s, kDruck^s, kFluss^s \rangle
\end{array}
$$

---

temporal

$$
\begin{array}{lll}
pH\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, pH\_korrektur, \Delta_{pH}, pH) \\
& & \qquad \land\ MaxTimer(t) \\
pO_2\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, pO_2\_korrektur, \Delta_{pO_2}, pO_2) \\
& & \qquad \land\ MaxTimer(t) \\
temp\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, reaktor.heizen(temp^s), \Delta_{temp}, temperatur) \\
& & \qquad \land\ MaxTimer(t) \\
drehz\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, reaktor.drehzahl(drehz^s), \Delta_{drehz}, ruehrerdrehzahl) \\
& & \qquad \land\ MaxTimer(t) \\
mDruck\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, reaktor.mDruck(mDruck^s), \Delta_{mDruck}, membrandruck) \\
& & \qquad \land\ MaxTimer(t) \\
kDruck\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, reaktor.kDruck(kDruckd^s), \Delta_{kDruck}, kopfraumdruck) \\
& & \qquad \land\ MaxTimer(t) \\
mFluss\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, reaktor.mFluss(mFluss^s), \Delta_{mFluss}, membranfluss) \\
& & \qquad \land\ MaxTimer(t) \\
kFluss\_richtig & \triangleq & \exists\, t : \land\ VTimer(t, reaktor.kFluss(kFluss^s), \Delta_{kFluss}, kopfraumfluss) \\
& & \qquad \land\ MaxTimer(t) \\
grenzBedingung & \triangleq & \land\ Min\_pO_2 \leq pO_2 \leq Max\_pO_2 \\
& & \land\ Min\_pH \leq pH \leq Max\_pH \\
& & \land\ Min\_temp \leq temperatur \leq Max\_temp \\
& & \land\ Min\_drehz \leq ruehrerdrehzahl \leq Max\_drehz \\
actions & \triangleq & \lor\ \exists\, s : pH\_soll(s) \\
& & \lor\ \exists\, s : pO_2\_soll(s) \\
& & \lor\ \exists\, s : temp\_soll(s) \\
& & \lor\ \exists\, s : drehz\_soll(s) \\
& & \lor\ \exists\, s : mDruck\_soll(s) \\
& & \lor\ \exists\, s : kDruck\_soll(s) \\
& & \lor\ \exists\, s : mFluss\_soll(s) \\
& & \lor\ \exists\, s : kFluss\_soll(s)
\end{array}
$$

$$
\begin{aligned}
spec \triangleq\ & \wedge\ Init \\
& \wedge\ pH\_richtig \\
& \wedge\ pO_2\_richtig \\
& \wedge\ temp\_richtig \\
& \wedge\ drehz\_richtig \\
& \wedge\ mDruck\_richtig \\
& \wedge\ kDruck\_richtig \\
& \wedge\ mFluss\_richtig \\
& \wedge\ kFluss\_richtig \\
& \wedge\ \Box grenzBedingung \\
& \wedge\ \Box[actions]_{sollVars}
\end{aligned}
$$

A.7 Modul Kultivierung

module *Kultivierung*

parameters

$bakterien, letzte : Nat^{0+}$ VARIABLE

$volumen : Real^{0+}$ VARIABLE

$AktiveZellen_{max}, \Delta_{doppel}, SichtAnzahl$: CONSTANT

import *MiscDef*

import *Echtzeit*

include *Umgebung* **as** *umgebung*

include *Medium* **as** *medium*

include *Bioreaktor* **as** *bioreaktor*

include *Kontrolle* **as** *kontrolle*

predicates

$Init \triangleq letzte = 0$

actions

$$verdoppelung \triangleq \begin{array}{l} \wedge\ aktiveZellen \geq 2 \cdot letzte \\ \wedge\ letzte' = aktiveZellen \\ \wedge\ \text{UNCHANGED}\ \langle bakterien, volumen \rangle \end{array}$$

temporal

$$ziel \triangleq \begin{array}{l} \wedge\ \exists\, x : \wedge\ Vtimer(x, verdoppelung, \Delta_{doppel}, letzte) \\ \qquad\qquad \wedge\ MaxTimer(x) \\ \qquad \wedge\ \Box[(\vee\ aktiveZellen \geq AktiveZellen_{max} \\ \qquad\qquad \vee\ bakterien \geq SichtAnzahl) \Rightarrow (\wedge\ \exists\, m, a, t : bioreaktor.entnahme(a, t, m) \\ \qquad\qquad\qquad\qquad\qquad\qquad \wedge\ volumen' = 0)]_{volumen} \end{array}$$

$$spec \triangleq \begin{array}{l} \wedge\ Init \\ \wedge\ umgebung.spec \\ \wedge\ medium.spec \\ \wedge\ bioreaktor.spec \\ \wedge\ kontrolle.spec \end{array}$$

Anhang B

Die Module zur Regelungssoftware

B.1 Modul Sensorprozess

module *Sensorprozess*

parameters
digWert, *bitWert*, *fBitWert*, *fVolt*, *fWert*, *trigger* : VARIABLE
MinBitWert, *MaxBitWert*, Δ, : CONSTANT
Kanal, *Verschiebung*, *Steigung* : CONSTANT

import *Echtzeit*
import *MiscDef*
import *Umrechnung*
include *A/D_Wandler* **as** *ADWandler* **with** *output* $\leftarrow$ *digWert*
include *Filter* **as** *filter* **with** *filterWert* $\leftarrow$ *fBitWert*

definitions
$anpassung(x) \triangleq (x + Verschiebung) \cdot Steigung$

actions

$$
\begin{array}{lll}
wertNeu & \triangleq & \wedge\ ADWandler.neuerWert(Kanal) \\
 & & \wedge\ bitWert' = min(max(MinBitWert, anpassung(digWert')), MaxBitWert) \\
 & & \wedge\ filter.filter(bitWert') \\
 & & \wedge\ fVolt' = bitToVolt(fBitWert') \\
 & & \wedge\ fWert' = bitToWert(fBitWert', Kanal) \\
 & & \wedge\ trigger' \neq trigger
\end{array}
$$

---

temporal

$$
\begin{array}{lll}
spec & \triangleq & \wedge\ \exists\, x : \wedge\ Vtimer(x, wertNeu, \Delta, trigger) \\
 & & \qquad\quad \wedge\ ExactTimer(x, wertNeu, trigger) \\
 & & \wedge\ \Box[wertNeu]_{\langle digWert, bitWert, fBitWert, fVolt, fWert, trigger \rangle}
\end{array}
$$

B.2 Modul Aktorprozess

module *Aktorprozess*

parameters
wert, *bitWert*, *volt* : VARIABLE
Kanal : CONSTANT

import *Umrechnung*
include *D/A_Wandler* **as** *DAWandler*

actions

$$
\begin{aligned}
putData \triangleq\ & \wedge\ bitWert' = wertToBit(wert, Kanal) \\
& \wedge\ volt' = bitToVolt(bitWert') \\
& \wedge\ DAWandler.DA\_Wandlung(bitWert') \\
& \wedge\ \text{UNCHANGED}\ wert
\end{aligned}
$$

temporal

$$spec \triangleq \Box[putData]_{\langle wert, bitWert, volt \rangle}$$

B.3 Modul Regelung

module *Regelung*

parameters

$pHStell, pO_2Stell, O_2Stell, N_2Stell, CO_2Stell, luft_1Stell$: $Real^{0+}$ VARIABLE
$trigger$: VARIABLE
$VStromSoll, CO_2Max, \Delta$: CONSTANT
$pO_2Regler$: $Regeltypen$ CONSTANT
$pHRegler$: $Regeltypen \setminus \{$"FUZZY"$\}$ CONSTANT
$Kanal_{O_2}, Kanal_{N_2}, Kanal_{CO_2}, Kanal_{luft_1}, Kanal_{pO_2}, Kanal_{pH}$: CONSTANT

---

import *Echtzeit*
include *Regler* **as** *regler* **with** $pH \leftarrow pHStell$
$pO_2 \leftarrow pO_2Stell$
include *Aktorprozess* **as** $O_2Aktor(Kanal_{O_2})$ **with** $wert \leftarrow O_2Stell$
include *Aktorprozess* **as** $N_2Aktor(Kanal_{N_2})$ **with** $wert \leftarrow N_2Stell$
include *Aktorprozess* **as** $CO_2Aktor(Kanal_{CO_2})$ **with** $wert \leftarrow CO_2Stell$
include *Aktorprozess* **as** $luft_1Aktor(Kanal_{luft_1})$ **with** $wert \leftarrow luft_1Stell$
include *Aktorprozess* **as** $pO_2Aktor(Kanal_{pO_2})$ **with** $wert \leftarrow pO_2Stell$
include *Aktorprozess* **as** $pHAktor(Kanal_{pH})$ **with** $wert \leftarrow pHStell$

---

definitions

$vars \triangleq \langle pHStell, pO_2Stell, O_2Stell, N_2Stell, CO_2Stell, luft_1Stell, trigger \rangle$
$Regeltypen \triangleq \{$"P", "PD", "PI", "PID", "FUZZY"$\}$

---

actions

$$\begin{aligned} neuberechnung \triangleq\ & \wedge\ regler.berechnepHStell(pHRegler) \\ & \wedge\ regler.berechnepO_2Stell(pO_2Regler) \\ & \wedge\ mischung \\ & \wedge\ putData \\ & \wedge\ trigger' \neq trigger \end{aligned}$$

$$\begin{aligned} mischung \triangleq\ & \wedge\ CO_2Stell' = ((CO_2Max - CO_2Min) \cdot (1 - \tfrac{pHStell'}{14})) + CO_2Min \\ & \wedge\ pO_2Stell' = \tfrac{4.63 \cdot O_2Stell' + Luft_1Stell'}{VStromSoll} \\ & \wedge\ O_2Stell' = \textbf{if}\ pO_2Stell' \leq \tfrac{VStromSoll - CO_2Stell'}{VStromSoll} \cdot 21.6\ \textbf{then}\ 0 \\ & \wedge\ N_2Stell' = \textbf{if}\ O_2Stell' \neq 0\ \textbf{then}\ 0 \\ & \wedge\ VStromSoll = CO_2Stell' + O_2Stell' + N_2Stell' + luft_1Stell' \end{aligned}$$

$$
\begin{aligned}
putData \triangleq\ & \wedge\ O_2Aktor.putData \\
& \wedge\ N_2Aktor.putData \\
& \wedge\ CO_2Aktor.putData \\
& \wedge\ luft_1Aktor.putData \\
& \wedge\ pO_2Aktor.putData \\
& \wedge\ pHAktor.putData
\end{aligned}
$$

---

temporal

$$
\begin{aligned}
spec \triangleq\ & \wedge\ \exists\, x : \wedge\ Vtimer(x, neuberechung, \Delta, trigger) \\
& \qquad\qquad \wedge\ ExactTimer(x, neuberechnung, trigger) \\
& \wedge\ \Box[neuberechnung]_{vars}
\end{aligned}
$$

B.4 Modul SoftwareKontrolle

module *SoftwareKontrolle*

import *InitKonstanten*
import *Echtzeit*
include *Sensorprozess* **as** $O_2Sensor(Min_{O_2}, Max_{O_2}, \Delta_{O_2}, Kanal_{O_2}, Vers_{O_2}, Steig_{O_2})$
include *Sensorprozess* **as** $N_2Sensor(Min_{N_2}, Max_{N_2}, \Delta_{N_2}, Kanal_{N_2}, Vers_{N_2}, Steig_{N_2})$
include *Sensorprozess* **as** $CO_2Sensor(Min_{CO_2}, Max_{CO_2}, \Delta_{CO_2}, Kanal_{CO_2}, Vers_{CO_2}, Steig_{CO_2})$
include *Sensorprozess* **as** $luft_1Sensor(Min_{luft_1}, Max_{luft_1}, \Delta_{luft_1}, Kanal_{luft_1}, Vers_{luft_1}, Steig_{luft_1})$
include *Sensorprozess* **as** $pO_2Sensor(Min_{pO_2}, Max_{pO_2}, \Delta_{pO_2}, Kanal_{pO_2}, Vers_{pO_2}, Steig_{pO_2})$
include *Sensorprozess* **as** $pHSensor(Min_{pH}, Max_{pH}, \Delta_{pH}, Kanal_{pH}, Vers_{pH}, Steig_{pH})$
include *Regelung* **as** $regelung(VStrom, CO_2Max, \Delta_{StellNeu}, pO_2Regler, pHRegler, Kanal_{O_2}, Kanal_{N_2}, Kanal_{CO_2}, Kanal_{luft_1}, Kanal_{pO_2}, Kanal_{pH})$

temporal

$$spec \triangleq \begin{array}{l} \wedge\ O_2Sensor.spec \\ \wedge\ N_2Sensor.spec \\ \wedge\ CO_2Sensor.spec \\ \wedge\ luft_1Sensor.spec \\ \wedge\ pO_2Sensor.spec \\ \wedge\ pHSensor.spec \\ \wedge\ regelung.spec \end{array}$$

Anhang C

Tabellen

Auf den folgenden Seiten sind die Tabellen mit den Zustandsvariablen- und Konstantenbezeichnern, sowie deren Bedeutung, sortiert nach Modulen abgebildet.

Modul	Zustandsvariable	Bedeutung
Sensorprozess	digWert	digitaler Meßwert vom A/D-Wandler
	bitWert	digitaler Meßwert nach der Kalibrierung
	fBitWert	gefilterter digitaler Meßwert
	fVolt	gefilterter Meßwert als Reglerspannung
	fWert	gefilterter Meßwert in physikalischer Einheit
	trigger	Triggert den Timer
Aktorprozess	wert	Stellwert in physikalischer Einheit
	bitWert	digitaler Stellwert für D/A-Wandler
	volt	Stellwert als Reglerspannung
Regelung	$pHStell$	Stellwert für pH
	pO_2Stell	Stellwert für pO_2
	O_2Stell	Stellwert für O_2
	N_2Stell	Stellwert für N_2
	CO_2Stell	Stellwert für CO_2
	$luft_1Stell$	Stellwert für Luftzufuhr der Membran
	trigger	Triggert den Timer

Tabelle C.1: Verzeichnis aller Zustandsvariablen aus der Softwarebeschreibung

Modul	Konstante	Bedeutung
Sensorprozess	MinBitWert	minimaler digitaler Meßwert
	MaxBitWert	maximaler digitaler Meßwert
	Δ	Zeitintervall zwischen zwei Messungen
	Kanal	Kanal des A/D-Wandlers
	Verschiebung	additive Konstante zur Meßwertverschiebung
	Steigung	Steigungsanpassung des Meßwertes
Aktorprozess	Kanal	Kanal des D/A-Wandlers
Regelung	VStromSoll	Volumenstromsollwert der Membran
	CO_2Max	Maximaler Anteil von CO_2 an der Membrangasmischung
	Δ	Zeitintervall zwischen zwei Stellwertberechnungen
	$pO_2Regler$	Einstellung der Reglerart z.B. PID oder Fuzzy für Luftsättigung
	$pHRegler$	Einstellung der Reglerart z.B. PID oder Fuzzy für pH
	$Kanal_{O_2} \ldots Kanal_{pH}$	Siehe Aktorprozess
SoftwareKontrolle	Modul InitKonstanten	Siehe Sensorprozess

Tabelle C.2: Verzeichnis aller Konstanten aus der Softwarebeschreibung

Modul	Zustandsvariable	Bedeutung
Echtzeit	now	Jetzt-Zeitpunkt
Umgebung	temperatur	Temperatur der Umgebung
	helligkeit	Helligkeit der Umgebung
Medium	aktiveZellen	Anzahl der lebenden Zellen im Medium
	toteZellen	Anzahl der toten Zellen im Medium
	bakterien	Anzahl der Bakterien im Medium
	pH	pH-Wert des Mediums
	pO_2	Luftsättigung des Mediums
	temperatur	Temperatur des Mediums
	raumtemperatur	aus Umgebung
	medium	Mediumszutaten und Giftstoffe als Funktion
Bioreaktor	ruehrerdrehzahl	Umdrehungszahl des Rührers
	membrandruck	Druck in der Begasungsmembran
	membrangas	Zusammensetzung der Gasmischung in der Membran
	membrangasfluss	Massendurchfluß durch die Begasungsmembran
	kopfraumdruck	Druck im Kopfraum
	kopfraumgas	Zusammensetzung der Gasmischung im Kopfraum
	kopfraumgasfluss	Massendurchfluß durch den Kopfraum
	volumen	Volumen des Kulturmediums
	temperatur	aus Medium
	aktiveZellen	aus Medium
	toteZellen	aus Medium
	bakterien	aus Medium
	medium	aus Medium
Kontrolle	pH	aus Medium
	pO_2	aus Medium
	temperatur	aus Medium
	ruehrerdrehzahl	aus Bioreaktor
	membrandruck	aus Bioreaktor
	membrangasfluss	aus Bioreaktor
	kopfraumdruck	aus Bioreaktor
	kopfraumgasfluss	aus Bioreaktor
	pH^s	Sollwert für pH
	pO_2^s	Sollwert für Luftsättigung
	$temp^s$	Sollwert der Mediumstemperatur
	$drehz^s$	Sollwert der Rührerdrehzahl
	$mDruck^s$	Sollwert für den Druck in der Begasungsmembran
	$mFluss^s$	Sollwert für den Massendurchfluß in der Membran
	$kDruck^s$	Sollwert für den Druck im Kopfraum
	$kFluss^s$	Sollwert für den Massendurchfluß im Kopfraum
Kultivierung	bakterien	aus Medium
	volumen	aus Bioreaktor
	letzte	Zeitpunkt der letzten Verdoppelung der Zellzahl

Tabelle C.3: Verzeichnis aller Zustandsvariablen aus der Reaktorbeschreibung

Modul	Konstante	Bedeutung
Echtzeit	∞	Unendlich (nie erreichbarer Zeitpunkt)
Umgebung	MaxHelligkeit	Obere Grenze für die Beleuchtung
Medium	Zutaten	Menge aller Inhaltsstoffe des Mediums
	Gifte	Menge aller Ausscheidungen der Zellen
	$Medium_0$	Initialwert der Mediumszusammensetzung
	pH_0	Initialwert pH
	$pO_{2(0)}$	Initialwert Luftsättigung
	$temperatur_0$	Initialwert Mediumstemperatur
	$AktiveZellen_0$	Initialwert lebende Zellen
	$ToteZellen_0$	Initialwert tote Zellen
Bioreaktor	Zutaten	s. Medium
	$Volumen_0$	Initialwert Volumen des Mediums
	$Drehzahl_0$	Initialwert Rührerdrehzahl
	$Membrandruck_0$	Initialwert Druck in Begasungsmembran
	$Membrangas_0$	Initialwert Zusammensetzung des Gases in der Membran
	$Membrangasfluss_0$	Initialwert Massendurchfluß in der Begasungsmembran
	$Kopfraumdruck_0$	Initialwert Druck im Kopfraum
	$Kopfraumart_0$	Initialwert Art der Kopfraumbegasung
	$Kopfraumgasfluss_0$	Initialwert Massendurchfluß im Kopfraum

Tabelle C.4: Verzeichnis der Konstanten aus der Reaktorbeschreibung, Module Echtzeit, Umgebung, Medium und Bioreaktor

Modul	Konstante	Bedeutung
Kontrolle	Min_pO_2	Min. Luftsättigung ohne Gefährdung der Kultivierung
	Max_pO_2	Max. Luftsättigung ohne Gefährdung der Kultivierung
	Min_pH	Min. pH ohne Gefährdung der Kultivierung
	Max_pH	Max. pH ohne Gefährdung der Kultivierung
	Min_temp	Min. Temperatur ohne Gefährdung der Kultivierung
	Max_temp	Max. Temperatur ohne Gefährdung der Kultivierung
	Min_drehz	Min. Rührerdrehzahl ohne Gefährdung der Kultivierung
	Max_drehz	Max. Rührerdrehzahl ohne Gefährdung der Kultivierung
	Δ_{pH}	Max. Zeitraum in dem pH vom Sollwert abweichen darf
	Δ_{pO_2}	Max. Zeitraum für Luftsättigung
	Δ_{temp}	Max. Zeitraum für Mediumstemperatur
	Δ_{drehz}	Max. Zeitraum für Rührerdrehzahl
	Δ_{mDruck}	Max. Zeitraum für Druck in der Membran
	Δ_{kDruck}	Max. Zeitraum für Druck im Kopfraum
	Δ_{mFluss}	Max. Zeitraum für Massendurchfluß in der Membran
	Δ_{kFluss}	Max. Zeitraum für Massendurchfluß im Kopfraum
	pH_0^s	Sollwert für pH
	$pO_{2(0)}^s$	Sollwert für Luftsättigung
	$temp_0^s$	Sollwert für Mediumstemperatur
	$drehz_0^s$	Sollwert für Rührerdrehzahl
	$mDruck_0^s$	Initialwert: Sollwert für Membrandruck
	$mFluss_0^s$	Initialwert: Sollwert für Massendurchfluß in der Membran
	$kDruck_0^s$	Initialwert: Sollwert für Druck im Kopfraum
	$kFluss_0^s$	Initialwert: Sollwert für Massendurchfluß im Kopfraum
Kultivierung	$AktiveZellen_{max}$	Max. Zellzahl der Kultivierung
	Δ_{doppel}	Verdoppelungszeit der Zellen
	SichtAnzahl	Bakterienanzahl damit Verseuchung erkannt wird

Tabelle C.5: Verzeichnis der Konstanten aus der Reaktorbeschreibung, Module Kontrolle und Kultivierung

Literaturverzeichnis

[ABL96] Jean-Raymond Abrial, Egon Börger, Hans Langmaack: Editors. Formal Methods for Industrial Applications – Specifying and Programming the Steam Boiler Control. Lecture Notes in Computer Science, Vol. 1165, Springer Verlag, 1996.

[AL92] Martín Abadi, Leslie Lamport. An Old-Fashioned Recipe for Real Time. ACM Transactions on Programming Languages and Systems, 16(5):1543-1571, September 1994.

[Bock96] Susanne Bockholt. Entwicklung eines digitalen Steuerungs- und Regelungssystems für Bioreaktoren. Diplomarbeit im Fach Naturwissenschaftliche Informatik, Arbeitsgruppe Zellkulturtechnik, Universität Bielefeld, 1996.

[Hen97] Dirk Henkel. Safely Sliding Windows. Technical Report RVS-Occ-97-05 der Technischen Fakultät, Arbeitsgruppe RVS, Universität Bielefeld. Available from `http://www.rvs.uni-bielefeld.de/`.

[Lüt93] Dirk Lütkemeyer. Mikroporöse Membranionenaustauscher zur schnellen Reinigung von Proteinen aus tierischen Zellkulturen. Dissertation an der Technischen Fakultät der Univeristät Bielefeld, Juli 1993.

[Lad95] Peter Ladkin. Formal But Lively Buffers in TLA$^+$. Technical Report 96-07 der Technischen Fakultät, Arbeitsgruppe RVS, Universität Bielefeld. Available from `http://www.rvs.uni-bielefeld.de/`.

[Lad96.1] Peter Ladkin. Lazy Cache Implements Complete Cache. Research Report RVS-RR-96-06 der Technischen Fakultät, Arbeitsgruppe RVS, Universität Bielefeld. Available from `http://www.rvs.uni-bielefeld.de/`.

[Lad97.1] Peter Ladkin. The Success and Failure of Artifacts. Technical Report RVS-Occ-97-04 der Technischen Fakultät, Arbeitsgruppe RVS, Universität Bielefeld. Available from `http://www.rvs.uni-bielefeld.de`.

[Lad97.2] Peter Ladkin. Computer-Related Incidents with Commercial Aircraft, Compendium. Arbeitsgruppe RVS, Universität Bielefeld. Available from `http://www.rvs.uni-bielefeld.de`.

[Lam93] Leslie Lamport. Hybrid Systems in TLA$^+$. Grossmann, Nerode, Ravn, Rischl: Editors. Hybrid Systems, Lecture Notes in Computer Science, Vol. 736. Springer Verlag, 1993.

[Lam94] Leslie Lamport. The temporal logic of actions. ACM Transactions on Programming Languages and Systems, 16(3):872-923, Mai 1994.

[Lam95] Leslie Lamport. TLA$^+$.
Available from `http://www.research.digital.com/SRC/tla/`.

[Lev95] Nancy Leveson. Safeware, Addison-Wesley 1995.

[MFJ97] A.Mokkedem, M. J. Ferguson, and R. deB. Johnston. A TLA Solution to the Specification and Verification of the RLP1 Retransmission Protocol. INRS-Telecommunications, Montreal, Quebec. Available from `http://www.inrs-telecom.uquebec.ca/users/mike/`.

[Pirt75] S. John Pirt. Principles of Microbe and Cell Cultivation. Oxford: Blackwell Scientific Publications, 1975.

[Sea96] John R. Searle. The Construction of Social Reality. London: Penguin Books, 1996.

Erklärung

Hiermit vesicherere ich, daß ich die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Bielefeld, den 11. Juli 1997

Lutz Sommerfeld