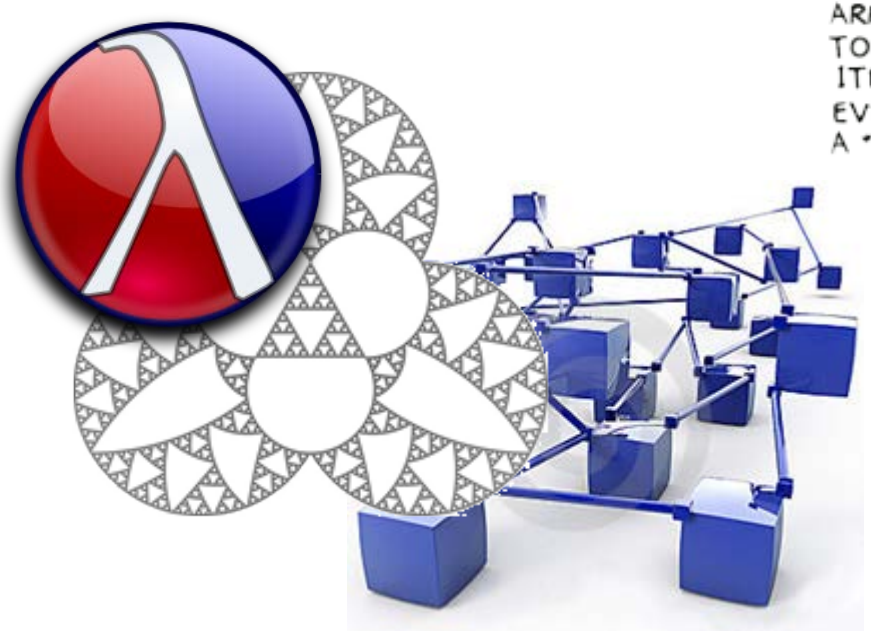
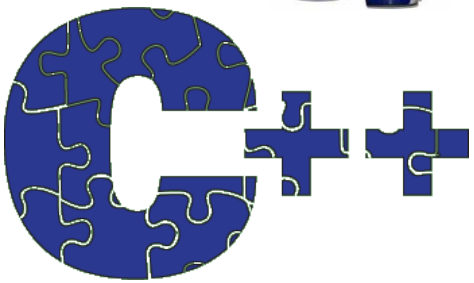
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Institut für Informatik





Informatik II – SS 2014
Einführung

Martin Butz, martin.butz@uni-tuebingen.de

Ein wenig über mich:
Curriculum Vitae

Werdegang

- Abitur am Gymnasium in Bad Königshofen i. Gr. 1995
  - Auslandsaufenthalt in Cape Town, SA (1992-1993)
- Studium der Informatik an der Universität Würzburg (1995-2001)
  - Nebenfach Mathematik (Vordiplom)
  - Nebenfach Psychologie (Diplom)
  - Diplom 2001
  - Auslandsaufenthalt in Urbana-Champaign, IL, USA (visiting scholar, 1999-2000)
- PhD Studium in Informatik an der University of Illinois at Urbana-Champaign, IL (2002-2004).
- Habilitation & Privatdozentur an der Universität Würzburg in Informatik (2010)
- Berufung nach Tübingen für die Professur „Kognitive Modellierung“ (2011)

Wissenschaftliche Arbeit

- Projekte
  - DFG Projekt: Modellierung von Antizipativer Kognition
  - EU Projekt MindRACES: From Reactive to Anticipatory Cognitive Systems
  - Emmy Noeter Projekt: Cognitive Bodyspaces: Learning and Behavior
  - Honda Research: Tracking and Anticipating the Behavior of other Cars
- Systeme
  - Learning Classifier Systems
    - Schema lernende Systeme (ACS2 ..)
    - XCS / XCSF system (datamining, regression learning)
  - Neural Networks
    - Growing Neural Gas (temporal learning)
    - Hebbian learning
    - Rekurrennte NNs (z.B. RNNPBs)
  - Bayesian Information Processing
  - SURE_REACH (model of human reaching)
  - Car racing games (adaptive bots)
- Aktuelle wissenschaftliche Arbeiten:
  - Intelligente, adaptive Roboter (in Simulationsumgebungen)
  - Modellierung des eigenen Körpers und der Interaktion mit bzw. Manipulation von Objekten
  - Wahrnehmung des eigenen Körpers im Raum
  - Embodiment (Sprache, Kognition, Verhalten)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Ein wenig über meinen Lehrstuhl: Lehrstuhl für Kognitive Modellierung

Mitarbeiter

- Dr. Anna Belardinelli
- Dr. Jan Kneissler

- Andreas Alin
- Stephan Ehrenfeld
- Johannes Lohmann
- Fabian Schrodt

Forschungsschwerpunkte

- Wie wir unseren eigenen Körper wahrnehmen
  - Multimodale Modelle
  - Probabilistisch & Neuronal
  - Maschinelles Lernen
  - Psychologische Experimente:
    - Rubber Hand Illusion, Virtual Reality, Eye-Tracking & Motion Tracking
- Wie wir unseren eigenen Körper kontrollieren
  - Antizipation und Zielorientiertheit
  - Auflösung von Redundanzen
  - Zielorientierte Suche nach Informationen
  - Psychologische Experimente
- Wie wir Lernen
  - Körpererkennung und –kontrolle
  - Objekterkennung und –manipulation
  - Abstraktere Kognition

--------- Und Sie ???? ---------

Studiengang…:

Informatik Bachelor

Medieninformatik Bachelor

Bioinformatik Bachelor

Medizininformatik Bachelor

Kognitionswissenschaftler

Lehramt-Informatik

Nebenfächler & Andere

Semester…:

Im
1ten Semester

Im
2ten Semester

In einem
höheren Semester

Herkunft…:

Aus
BW – Baden-Württemberg

Aus
Anderes Bundesland in D.

Aus
Einem anderen Land

Heute: Einführung

- Was wird hier gelehrt?
- Organisatorisches
- Literatur
- Allgemeine Hinweise
- Motivation und Einordnung

- Scheme & Dr. Racket: Erste Programmbeispiele

Informatik II Lehren – was genau?

Informatik II lehren bedeutet (für mich) … :

- **Perspektiven und Sichtweisen vermitteln**
  - Wie komme ich von einem Problem zu einem sinnvollen Programm?
- **Prinzipien und Herangehensweisen aufzeigen**
  - Welche Prinzipien unterliegen verschiedenen Programmiersprachen?
  - Wie gehe ich ein Problem entsprechend an?
- **Unterschiedliche Denkweisen kontrastieren**
  - Programmiersprachen forcieren (sehr) strukturiertes Denken.
  - Wie unterscheiden sich diese Strukturen in den verschiedenen Programmiersprachen?
- **Zum Reflektieren ermutigen**
  - Was habe ich jetzt wirklich gelernt? … Programmieren? Bestimmte Strukturen und Konzepte? Was ist ein Problem?

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Ziel der Vorlesung: Programmierkonzepte Verstehen

- **Kontrastierung zweier fundamentaler Programmierkonzepte:**
  - **Funktionale Programmierung**
    - Programme sind Funktionen
    - Daten sind Werte, komplexe Strukturen oder Funktionen
    - Rekursion als eines der wichtigsten Programmierkonzepte
    - Funktionen als Übergabeparameter
    - Schablonen zur flexiblen Abarbeitung von Datenstrukturen
    - Programmiersprach: Scheme
  - **Imperative Programmierung**
    - Programme sind Abläufe
    - Daten sind Werte, Zeiger oder komplexe Strukturen
    - Methoden und Klassen als wichtigste Programmierkonzepte
    - Werte oder Zeiger als Übergabeparameter
    - Klassenstrukturen für die strukturierte Bearbeitung von Datenstrukturen
    - Programmiersprache: C++
- **Außerdem:**
  - Was ist überhaupt ein Programm?
  - Vom Problem zum Programm
  - Programmdesign

ORGANISATORISCHES

- Ablauf der Semesters
- Personen
- Organisation der Übungen & Übungsgruppen
- Abgabe von Übungen
- Benotung

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Das Semester auf einen Blick
(Änderungen vorbehalten)

Woche 1 (KW 15, 07.04. - 13.04.):
- VL1: VL Überblick & Hintergrund – Dr. Racket Tool
- VL2: [Prof. Küchlin Informatik Begrüßung] Scheme: Einfache Ausdrücke, Bindung mit define, lamda, …
- Ausgabe der PÜ-Scheme Einstiegs-Aufgaben (ohne Benotung)

Woche 2 (KW16, 14.04. - 20.04.):
- TÜ: Einführung in Scheme und Dr. Racket / Feedback, Fragen & Antworten zur VL
  PÜ: Erste Woche PÜs… Einarbeitung, Test-Abgabe, Installationsfragen
- VL3: Scheme: Abstraktion, Kommentare, Verträge
- VL4: Scheme: Konstruktionsanleitung für Prozeduren
- Ausgabe der 1ten PÜ Aufgaben

Woche 3 (KW17, 21.04. – 27.04.):
- TÜ: entfällt
  PÜ: **1. PÜ Abgabe** während der PÜ.
- VL5: Scheme: Fallunterscheidungen, Spezialform cond
- VL6: Scheme: Records und zusammengesetzte Daten
- Ausgabe des 1ten TÜ Übungsblatts

Woche 4 (KW18, 28.04. - 04.05.):
- TÜ und PÜ Veranstaltungen entfallen diese Woche ☺
- VL7: Scheme: Prozeduren über zusammengesetzte Daten
- **Abgabe 1. TÜ** Übungsblatt
  Ausgabe des 2ten TÜ Übungsblatts
  Ausgabe der 2ten PÜ Aufgaben

Woche 5 (KW19, 05.05. - 11.05.):
- TÜ: Besprechung 1. Übungsblatt
  PÜ: **2. PÜ Abgabe** während der PÜ.
- VL8: Scheme: Zusammengesetzte und gemischte Daten und Prozeduren dazu
- VL9: Scheme: Rekursion auf Listen, Konstruktionsanleitung für Listen-Prozeduren
- **Abgabe 2. TÜ** Übungsblatt
  Ausgabe des 3ten TÜ Übungsblatts.
  Ausgabe der 3ten PÜ Aufgaben

Woche 6 (KW20, 12.05. - 18.05.):
- TÜ: Besprechung 2. Übungsblatt
  PÜ: **3. PÜ Abgabe** während der PÜ.
- VL10: Scheme: Rekursion über natürliche Zahlen, Endrekursion & Reverse,
- VL11: Scheme: Higher-Order Prozeduren, Listenfaltung
- **Abgabe 3. TÜ** Übungsblatt
  Ausgabe der 4ten PÜ Aufgaben

Woche 7 (KW21, 19.05. - 25.05.):
- TÜ: Besprechung 3. Übungsblatt
  PÜ: **4. PÜ Abgabe** während der PÜ.
- VL12: Scheme: Anwendungen von HOP
- VL13: Das Lambda-Kalkül und einfach Grammatiken
- Ausgabe des 4ten TÜ Übungsblatts.

Woche 8 (KW22, 26.05. - 01.06.):
- TÜ und PÜ Veranstaltungen entfallen diese Woche ☺
- VL14: C++ Einführung: Datentypen, Zuweisungen und Kontrollfluss.
- **Abgabe 4. TÜ** Übungsblatt
  Ausgabe der PÜ-C++ Einstiegs-Aufgaben (ohne Benotung)

Woche 9 (KW23, 02.06. - 08.06.):
- TÜ: Besprechung 4. Übungsblatt & Einführung in C++
  PÜ: C++ Einarbeitung, Test-Abgabe, Installationsfragen
- VL15: C++: Bedingte Anweisungen und Schleifen
- VL16: C++: Arrays und Speicherverwaltung
- Ausgabe des 5ten TÜ Übungsblatts.

########## --------- **KW 24: Vorlesungsfrei - Pfingsten** --------- ######

Woche 10 (KW25, 16.06. - 22.06.):
- TÜ und PÜ Veranstaltungen entfallen diese Woche ☺
- VL17: C++: Funktionen
- **Abgabe 5. TÜ** Übungsblatt
  Ausgabe des 6ten TÜ Übungsblatts.
  Ausgabe der 5ten PÜ Aufgaben

Woche 11 (KW26, 23.06. - 29.06.):
- TÜs: Besprechung 5. Übungsblatt
  PÜs: **5. PÜ Abgabe** während der PÜ.
- VL18: C++: Klassen und Objekte
- VL19: C++: Objektorientierte Programmierung II
- **Abgabe 6. TÜ** Übungsblatt
  Ausgabe des 7ten TÜ Übungsblatts.
  Ausgabe der 6ten PÜ Aufgaben

Woche 12 (KW27, 30.06. - 06.07.):
- TÜs: Besprechung 6. Übungsblatt
  PÜs: **6. PÜ Abgabe** während der PÜ.
- VL20: C++: Funktionszeiger
- VL21: C++: Templates
- **Abgabe 7. TÜ** Übungsblatt
  Ausgabe der 7ten PÜ Aufgaben

Woche 13 (KW28, 07.06. - 13.06.):
- TÜ: Besprechung 7. Übungsblatt & Fragen zur Klausur und allgemein.
  PÜ: **7te PÜ Abgabe** während der PÜ.
- VL22: C++: Templates II & Hashing
- VL23: C++: Wiederholung und Zusammenfassung

Woche 14 (KW29, 14.06. - 20.06.): Keine Veranstaltungen – Zeit für die Vorbereitung der Klausur

Woche 15 (KW30, 21.06. - 27.06.): **KLAUSUR am 24.07.2014, 16:00-19:00 Uhr. Hörsäle N06, N07 & N08**

Woche 16 (KW 41, 06.10. - 12.10.): **NACH_KLAUSUR am 09.10.2014, 16:00-19:00 Uhr. Hörsäle: N06 & N07**

Dozent und Organisatoren

- Dozent der Vorlesung

  Prof. Martin Butz, Kognitive Modellierung
  Raum: Sand 14, C415
  email: martin.butz@uni-tuebingen.de

- Verantwortlicher Mitarbeiter

  Dr. Jan Kneissler
  Raum: Sand 14, C420
  email: jan.kneissler@uni-tuebingen.de

TutorInnen

1.	Blöck	Alexander	alexander.bloeck@student.uni-tuebingen.de
2.	Hofmeister	Julian	julian.hofmeister@student.uni-tuebingen.de
3.	Hohloch	Jan-Peter	jan-peter.hohloch@student.uni-tuebingen.de
4.	Hupp	Lena	lena-sophie.hupp@student.uni-tuebingen.de
5.	Jugl	Niklas	niklas.jugl@student.uni-tuebingen.de
6.	Kummle	Oliver	oliver.kummle@student.uni-tuebingen.de
7.	Ludwig	Thorsten	thorsten.ludwig@student.uni-tuebingen.de
8.	Merling	Daniel	daniel.merling@student.uni-tuebingen.de
9.	Mesaric	Robin	robin-marco.mesaric@student.uni-tuebingen.de
10.	Ploeger	Jannis	jannis.ploeger@student.uni-tuebingen.de
11.	Reisenauer	Matthias	matthias.reisenauer@student.uni-tuebingen.de
12.	Richter	Peter	peter.richter@student.uni-tuebingen.de
13.	Roehm	Yves	yves.roehm@student.uni-tuebingen.de
14.	Rosenkranz	Vinzenz	vinzenz.rosenkranz@student.uni-tuebingen.de
15.	Schulz	Cornelia	co.schulz@student.uni-tuebingen.de
16.	Schulz	Patrick	patrick.schulz@student.uni-tuebingen.de
17.	Stellmach	Hanna	hanna-stefanie.stellmach@student.uni-tuebingen.de
18.	Stockmayer	Andreas	andreas.stockmayer@student.uni-tuebingen.de
19.	Sahin	Kaan	kaan.sahin@student.uni-tuebingen.de
20.	Vial	Jens	jens.vial@student.uni-tuebingen.de
21.	Wilk	Kay	kay.wilk@student.uni-tuebingen.de

Webseiten

Arbeitsgruppe Kognitive Modellierung

http://www.cm.inf.uni-tuebingen.de

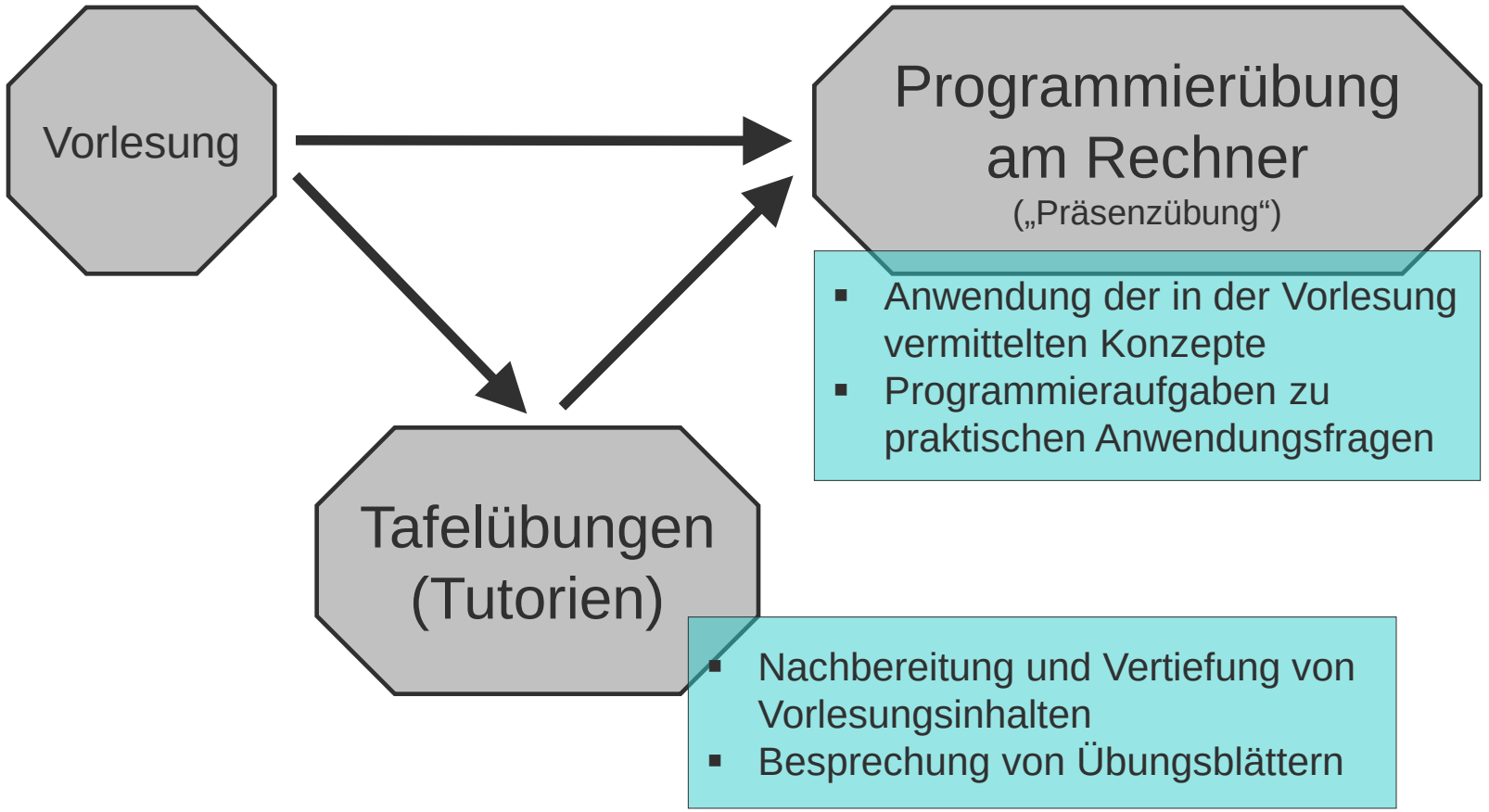
(-> Teaching -> Vorlesung: Informatik II)
http://www.cm.inf.uni-tuebingen.de/teaching/teaching-overview/sose-2014/vorlesung-informatik-ii.html

Lernplattform **CIS** (**Anmeldung bitte umgehend !!!**)

https://cis.informatik.uni-tuebingen.de/info2-ss-14

Vorlesung und Übungen

- Präsentationen (Laptop, Projektor, Tafel, Code, Programm…)
- Unterlagen: Vorlesungsmaterialien & Übungsaufgaben



Tutorien und Rechnerübungen

- Max. Anzahl an Studierenden pro Tafelübungsgruppe: 30
- Max. Anzahl an Studierenden pro Programmierübungsgruppe: 42 (verteilt in 3 Räume)
- Anmeldung / Vergabe über die “**CIS**” **Online Platform**

Grundsätzliches:

- Die Übungen sind integraler Bestandteil der Veranstaltung (!).
- Die Themen der Vorlesung sollen dort anhand **praktischer Beispiele** eingeübt und vertieft werden.
- In dieser Veranstaltung werden die **Grundlagen** für viele weitere Details für das spätere Informatikstudium gelegt.
- Alle Inhalte der Veranstaltung (Vorlesung, Tafelübung, Programmierübung) sind **klausurrelevant.**

Übungen – Ziele und Methodologie

- Vertiefen und Einüben des **Vorlesungsstoffes**
- Erlangen **praktischer Fertigkeiten** im Umgang mit einer Programmiersprache und Anwenden der **Konzepte** aus der Vorlesung:
  - Analyse und Strukturierung von Problemstellungen
  - Entwurf von Algorithmen
  - Entwicklung und Verstehen von Algorithmen
  - Erstellung von Programmen
  - Programmieren in **Scheme** bzw. **C++**

Organisation und Durchführung

- Alle Materialien zu den Übungen werden über die Plattform **CIS** bereit gestellt.
- Die „Tafelübungen“ werden in Tutorien durchgeführt.
- Es werden ca. 20 Tafelübungen mit je maximal 30 Teilnehmer/innen angeboten.
- **Einschreiben** für die Übungen:
  - Bitte melden Sie sich (falls noch nicht geschehen) noch HEUTE im **CIS** an!
  - Dann wählen Sie Ihre Übungstermine:
    1. Programmierübung: Bitte wählen sie als erstes einen noch verfügbaren Slot für Ihre PÜ.
       - Prinzip ist First-Come-First-Serve!
    2. Tafelübung: Bitte wählen Sie dann **drei** Wunschzeiten aus!
       - Die Zuweisung erfolgt in den nächsten Tagen (mittels eines Verteilungsalgorithmus, der die Präferenzen bestmöglich berücksichtigt).

https://cis.informatik.uni-tuebingen.de/info2-ss-14

CIS – Materialien und Veranstaltungsunterlagen

- Alle Vorlesungsmaterialien und Informationen zur Vorlesung und den Übungen werden wir über die Plattform **CIS** verwalten.

- Materialien
  - Vorlesungsfolien
  - Übungsblätter
  - Zusatzmaterialien / Handouts
  - Wiki
  - Forumsbereich: https://forum.informatik.uni-tuebingen.de/

- Auch zum Abgeben der Übungsblätter (**immer elektronisch im CIS**)!

- Tafelübungsblätter und auch Programmierübungsaufgaben werden immer im CIS **elektronisch veröffentlicht**.

Bearbeitung der Übungsblätter

- Tafelübungsblätter werden über das CIS ausgegeben und abgegeben.
  - Bearbeitungszeit ist typischerweise eine Woche.

- Programmieraufgaben werden auch über das CIS ausgegeben.
  - Ausgabe entweder direkt in der PÜ oder die Woche vorher im CIS.
  - Die Abgabe von Programmieraufgaben erfolgt über das **CIS** in den Präsenzübungen am Rechner.

Zusammenarbeit und Abgabe…

- Abgabe der Übungsblätter alleine oder in **Zweiergruppen**
- Abgabe der Programmierübung (am Rechner) **individuell**
  - ➢ **Zur Vorbereitung der Lösungen zu den Übungsblättern / der Programmierübungen kann und soll kooperiert und kommuniziert werden.**
- **Abgaben dürfen nicht identisch sein oder kopiert und modifiziert werden.**
- **Plagiate** werden geahndet!
  - Wir werden **alle** Abgaben auf Ähnlichkeit prüfen.
  - Alle beteiligten Personen bekommen **NULL Punkte.**
  - **Wiederholer** werden von der Veranstaltung ausgeschlossen.
- **Logins** dürfen nur von der **zugehörigen Person** genutzt werden.
  - **Passwörter werden nicht veröffentlicht.**

Übungsblätter

- Bearbeitung
  - Die Blätter werden grundsätzlich in 2er-Teams bearbeitet (alleine geht natürlich auch).
  - Teams sollten im gleichen Tutorium sein.
  - Besprechung der Aufgaben und der Musterlösung wird in den Tutorien stattfinden.

- Tafelübungen
  - Es wird eine aktive Teilnahme an den Übungen und in den Tutorien erwartet.
  - In jeder Übungsgruppe wird determiniert, wer bereit ist, eine Aufgabe vorzustellen.
  - Es stellt immer eine Übungsgruppe aus dem Tutorium seine Lösung vor.
  - Musterlösungen werden vom Tutor vorgestellt.

Abgabe der Lösung

- Die Abgabe einer Lösung bedeutet,
  - dass man diese verstanden hat und
  - in der Lage ist, diese auch in der Tafelübung zu präsentieren.

- Punkte für eine Aufgabe werden dann erreicht (und somit angerechnet), wenn
  - die Aufgabe erfolgreich bearbeitet und rechtzeitig abgegeben wurde,
  - man sich im Tutorium aktiv beteiligt und seine Lösung ggf. präsentiert.

Programmier- bzw. Präsenzübung:

- Praktische Programmieraufgaben
- Kleinere Programme für die Konzeptvertiefung
- Zusätzlicher Code in einem gegebenen Programm

- Ausgabe über CIS die Woche vorher **oder** erst direkt in der Präsenzübung

- Abgabe der Lösung in den Rechnerpools
  - Zeit 90 Minuten
  - Abgabe nur am zugewiesenem Rechner
- Lösungen der Aufgaben werden direkt in der Programmierübung **individuell** abgegeben und bewertet.

- Zu den Programmieraufgaben, die vorher ausgegeben werde:
  - Veröffentlichung im CIS – Nachricht im News-Bereich
  - **Vorbereitung zu Hause und in Gruppen ist dadurch möglich!**
  - In der Programmierübung selbst wird aber absolut ALLEINE gearbeitet.

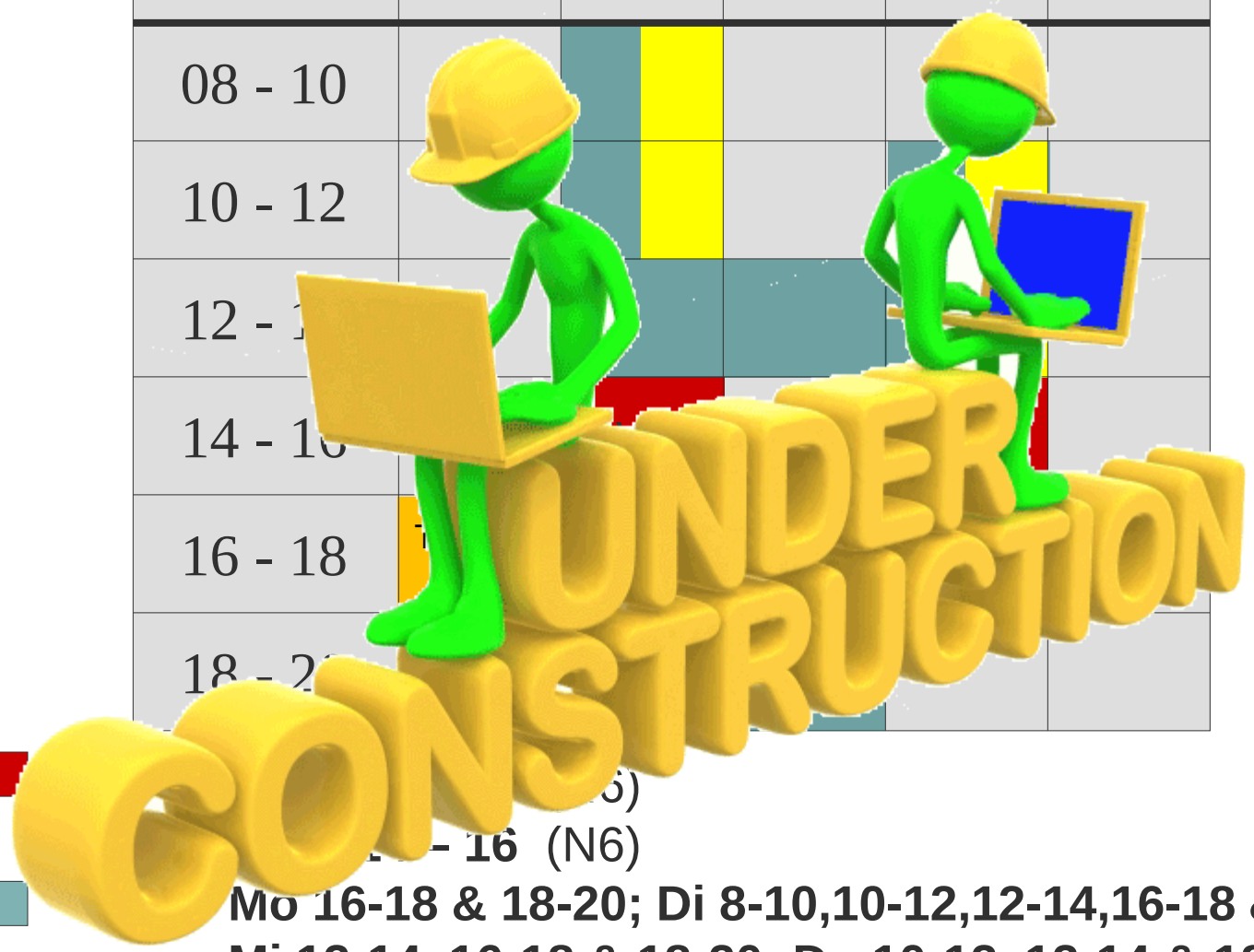
PÜs und TÜs Anwesenheitspflicht.

- TÜs:
  - Anwesenheitspflicht bei TÜ Blätter Besprechungen – sonst 0 Punkte.
  - Bei Krankheit: Attest einreichen & TÜ TutorIn informieren.

- PÜs:
  - Programme können nur selbst abgegeben werden.
  - Bei Krankheit:
    - PÜ Termin freigeben (im CIS möglich ab Sonntag 12:00 Uhr für die kommende Woche).
      - (In der Woche danach habt Ihr übrigens automatisch wieder Euren alten Standardtermin für die PÜ.)
    - Attest einreichen

- Das Attest:
  - Im Original oder eingescannt per email bei Frau Di Paolo, Raum C408, Sand 14 vorgelegt werden.
    - Sprechzeiten sind Mo-Do 10-12 Uhr und Mo 14-16 Uhr.
    - (Es kann eine Kopie erstellt werden, falls Sie das Original wieder mitnehmen müssen.)
  - Oder eingescannt per email an Frau Di Paolo: dipaolo@informatik.uni-tuebingen.de
  - Oder per Post (z.Hd. Frau Di Paolo, Raum C408, Sand 14, Tübingen 72076)
    - (Das Attest wird nicht zurückgeschickt werden.)

Zeit	MO	DI	MI	DO	FR
08 - 10					
10 - 12					
12 - 1					
14 - 16					
16 - 18					
18 - 2					



– 16 (N6)

Mo 16-18 & 18-20; Di 8-10,10-12,12-14,16-18 & 18-20; Mi 12-14, 16-18 & 18-20; Do 10-12, 12-14 & 18-20

TÜ **Di 8-10,10-12 & 18-20; Mi 17-19; Do 10-12,12-14 & 18-20**

(PÜ=„Programmier- bzw. Präsenzübung“
TÜ=„Tafelübung“)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Aktueller Plan (08.04.2014 13:00

Programmier- bzw. Präsenzübungen

- Bitte pünktlich zur vollen Stunde im Raum H33 im C-Bau zum Einteilen erscheinen (Im CIS ist aus technischen Gründen eine spätere Uhrzeit angegeben:
  - z.B. CIS sagt: „PÜ beginnt um 16:15" → um 16:00 s.t. Im H33 melden)
- Bei organisatorischen Fragen den PÜ Hauptverantwortlichen (Zeile 1) anschreiben! Email Liste der Tutoren im CIS Downloadbereich.

#	Di8	Di10	Di12	Di16	Mi8	Mi12	Mi14	Mi16	Do10	Do12	Do16
1	*Jan-Peter*	*Jan-Peter*	*Jan-Peter*	*Jan-Peter*	*Cornelia*	*Hanna*	*Vinzenz*	*Jannis*	*Cornelia*	*Julian*	*Niklas*
2	Lena	Hanna	Julian	Lena	Hanna	Julian	Yves	Vinzenz	Lena	Lena	Julian
3	Jannis	Daniel	Niklas	Daniel	Jannis	Oliver	Hanna	Oliver	Jens	Niklas	Cornelia
4	Daniel	Yves	Cornelia	Vinzenz	Jens	Jens	Oliver	Niklas		Oliver	Jens
5	Yves		Jannis		Daniel	Vinzenz				Yves	

Tafelübungen (beginn immer c.t.)

TU	Di8	Di10	Di16	Mi12	Mi17	Do12	Do16
1	Peter VBN3	Kaan AM04	Alexander N8	Robin N2	Andreas N9	Kay A3M04	Kaan A3M04
2	Patrick 1B01		Andreas VBN3		Kay VBN3	Robin A6G07	Patrick A6G07
3			Matthias 7E02			Peter N8	Alexander N5

Übungsgruppenanmeldung bitte bis MORGEN (MITTWOCH)

Arbeitsgruppe Kognitive Modellierung
http://www.cm.inf.uni-tuebingen.de
(-> Teaching -> Vorlesung: Informatik II)
http://www.cm.inf.uni-tuebingen.de/teaching/teaching-overview/sose-2014/vorlesung-informatik-ii.html

Lernplattform **CIS** (**Anmeldung bitte umgehend !!!**)
https://cis.informatik.uni-tuebingen.de/info2-ss-14

Hinweise:
Bis morgen 12:00 Uhr muss BITTE jeder Teilnehmer an der Vorlesung:

1. sich im CIS angemeldet haben: https://cis.informatik.uni-tuebingen.de/info2-ss-14/
2. unter "Programmierübungen -> Präsenzübung" einen PÜ Termin fürs ganze Semester auswählen
3. unter "Tafelübungen -> Gruppe auswählen" 3 Präferenzen für Übungsgruppen angeben (nein, weniger angeben geht nicht), darf natürlich nicht mit dem PÜ Termin überlappen!

Weitere Hinweise

- Achtung!!! : Wer sich bereits für eine PÜ eingetragen hat… es gab folgende **Änderungen**:
  - Die Gruppe Donnerstag 8-10 entfällt wegen zu wenig Nachfrage.
    - ➢ Bitte auf einen **anderen Termin** ausweichen!
- Es gibt drei zusätzliche Gruppen: Di10-12, Mi14-16, Do10-12
  - Wer möchte, kann gerne auf eine dieser Gruppen wechseln.
  - Wie funktioniert das Wechseln?
    - Im CIS unter "Programmierübungen -> Präsenzübung" hinter dem Termin auf
      - "Termin löschen drücken",
      - dann "Weiter",
      - dann "Fertigstellen"(wichtig!)
      - dann neuen Termin auswählen,
      - dann "Weiter„,
      - dann "Fertigstellen" (wieder sehr wichtig)

Benotung und Prüfungen

- SWS: 4 V + 4 Ü (8 ECTS)
- Tutorien: 25% der Note
- Rechnerübungen: 25% der Note
- Klausur 50% der Note
  - Klausurtermin: **24.07.2014** (Nachklausur 09.10.2014)

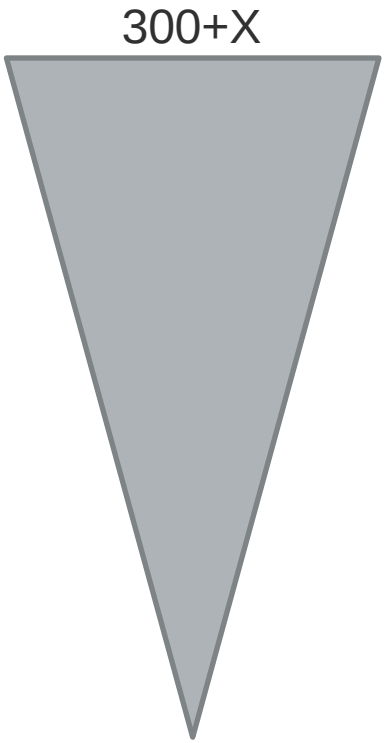
  - Die **Note** wird am Ende durch die Gewichtung der jeweils erreichten Prozente der jeweils möglichen Punkte ermittelt.
  - Die erreichten (gewichteten) Prozente werden dann in die Note umgerechnet.
  - Bestanden hat man
    - Wenn man deutlich über 50% der gewichteten Prozente erreicht hat **und**
    - Wenn man mindestens 40% der Punkte in der Klausur erreicht hat.

Benotung (Details)

- Übungsblätter: **25% der Punkte**
- Programmierübungen: **25% der Punkte**
- Klausur: **50% der Punkte**
  (Neu: Bestehhürde bei 40% - d.h. 40% der Punkte in der Klausur müssen erreicht werden, um den Kurs zu bestehen.)

- Beispiel:
  - StudentIn X hat erreicht:
    - PÜ: 23 von 85 zu erreichenden Punkten.
    - TÜ: 95 von 155 zu erreichenden Punkten.
    - Klausur: 66 von 100 zu erreichenden Punkten.
    - Deswegen hat der(ie) StudentIn:
      .25*23/85+ .25*95/155+ .5*66/100 = .25*.271+ .25*.613+ .5*.66 =
      = .551 (knapp bestanden – Note befriedigend bis ausreichend).
  - StudentIn Y hat erreicht:
    - PÜ: 82 von 85 zu erreichenden Punkten
    - TÜ: 149 von 155 zu erreichenden Punkten
    - Klausur: 55 von 100 zu erreichenden Punkten.
    - Deswegen hat der(ie) StudentIn:
      .25*82/85+ .25*149/155+ .5*55/100 = .25*.965+ .25*.961+.5*.55 =
      = .757 (klar bestanden – Note gut)

Bei Problemen / Fragen / Sorgen / Vorschlägen / Kritik etc…



Bitte an folgende Reihenfolge halten:

1. Tutor / Tutorin Ihrer Übungsgruppe
2. Forum im **CIS**
3. Übungsleiter: Jan Kneissler
4. Dozent: Martin Butz

(… sonst könnte die Reaktionszeit unnötig lang werden)

LITERATUR

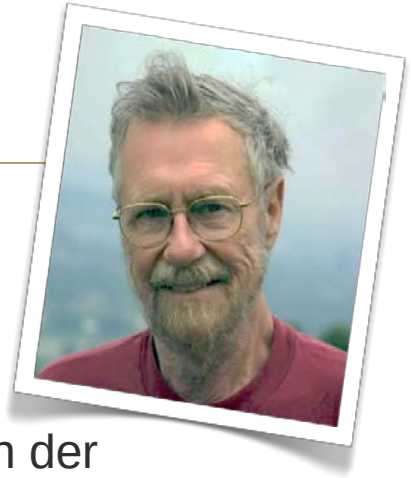
Bücher und Online Ressourcen für Scheme

- Die Macht der Abstraktion – Klären & Sperber
  - Sehr verständlich und zugänglich geschriebene Einführung.
  - Auch in der Bibliothek ausleibar.

- Alternative:
  - How to Design Programs - http://www.htdp.org/
  - Neuste Edition ist auch online: http://www.ccs.neu.edu/home/matthias/HtDP2e/

- Viele Ressourcen auch direkt in Dr. Racket!

INFORMATIK II: EINFÜHRUNG



Informatik = Computer Science

- Die Wissenschaft (digitaler) Rechnersysteme…
  "In der Informatik geht es genauso wenig um Computer wie in der Astronomie um Teleskope." (Edsger W. Dijkstra)

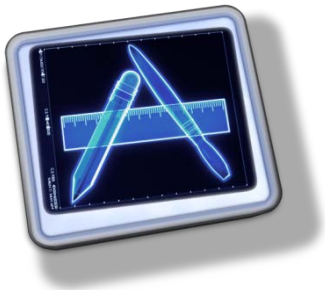
- Informatik: Information + Mathematik + Elektrotechnik
  - Die Wissenschaft computer–unterstützten Rechnens
  - Die Wissenschaft der systematischen, computer-unterstützten Informationsverarbeitung

- (seit ca. 1967; informatique, informatika, informatyka, informatics)

Informatik = (mindestens) **3-mal Informatik**

Praktische Informatik
- Programmierung, Systeme, Programmstrukturierung, Tools
  - "Berechnet diese Funktion eigentlich auch für *z = 0* das korrekte Ergebnis?"
  - „Wie konstruiere ich eine nützliche Programmiersprache?"

Theoretische Informatik
- Mathematik (Algebra, Logik), Berechenbarkeitsmodelle, Komplexität
  - "Wie langen müssten wir rechnen, wenn die Eingabe 100 x länger würde?"
  - „Ist dieses Problem theoretisch überhaupt lösbar?"



Technische Informatik
- Rechnerarchitektur, Bauteile, Netzwerkkommunikation, Hardware-Schnittstellen
  - "Wie können wir verhindern, dass ein defekter Speicherbaustein unbemerkt seinen Inhalt verändert?"
  - „Wie übertrage ich Daten Wireless am effektivsten?"

Logos © Apple Inc.

4te mal Informatik = Informatik als anwendungsorientierte Wissenschaft

- Informatik & Software ist überall.

- Informatik ist durch die Anwendung sehr interdisziplinär
  - Medieninformatik
  - Bioinformatik
  - Medizininformatik
  - Kognitionswissenschaft
  - Informatik-basierte (Computational) Neurowissenschaft

- Themen sind weitverbreitet und meist bekannt, wie z.B.
  - Datamining
  - (Prozess-) Modellierung
  - Suchmaschinen
  - Netzwerke (insbesondere auch sozial)
  - Robotik
  - …

Informatik = Systematische Problemlösung

Erstes Beispiel: Ein Parkplatz-Problem

- “Auf einem Parkplatz stehen PKWs und Motorräder (ohne Beiwagen).
  Zusammen seien es **n** Fahrzeuge
  mit insgesamt **r** Rädern.“
- Bestimme die Anzahl **p** der PKWs bzw.
  die Anzahl **m** der Motorräder.”

- Dies beschreibt gleich eine ganze Menge von Problemen, für jede Wahl von n und r.

- Also haben wir es mit den folgenden Funktionen zu tun:
  - p = P(n,r) Funktion P zur Berechnung der Anzahl der PKWs
  - m = M(n,r) Funktion M zur Berechnung der Anzahl der Motorräder

Das Parkplatz-Problem

- “Auf einem Parkplatz stehen PKWs und Motorräder (ohne Beiwagen).
  Zusammen seien es **n** Fahrzeuge mit insgesamt **r** Rädern.“
- Bestimme die Anzahl **p** der PKWs bzw. die Anzahl **m** der Motorräder.”

n = Anzahl der Fahrzeuge
r = Anzahl der Räder
p = P(n,r) = Anzahl der PKWs
m = Anzahl der Motorräder

- Mathematisch formuliert:
  - (I): $p + m = n$ bzw. $m = n - p$
  - (II): $4p + 2m = r$

  Ersetze m in II mit I…

  - $4p + 2 (n – p) = r$
  - $4p + 2n – 2p = r$
  - $2p = r – 2n$
  - $p = .5 (r – 2n)$

- Somit: $P(n,r) = .5 (r – 2n)$

n = Anzahl der Fahrzeuge
r = Anzahl der Räder
p = P(n,r) = Anzahl der PKWs
m = Anzahl der Motorräder

Kein Parkplatz-Problem mehr?

- Funktion als allgemeine Lösung sieht ja erst mal gut aus…
  Anzahl der PKWs p = P(n,r) = .5 (r – 2n)

- Oder?
- Überprüfung an Beispielen:

- P(3, 9) = 1.5	Halbe PKWs !?
- P(5, 2) = -4	Negative Anzahl an PKWs !?
- P(2, 10) = 3	Mehr PKWs als Fahrzeuge !?

n = Anzahl der Fahrzeuge
r = Anzahl der Räder
p = P(n,r) = Anzahl der PKWs
m = Anzahl der Motorräder

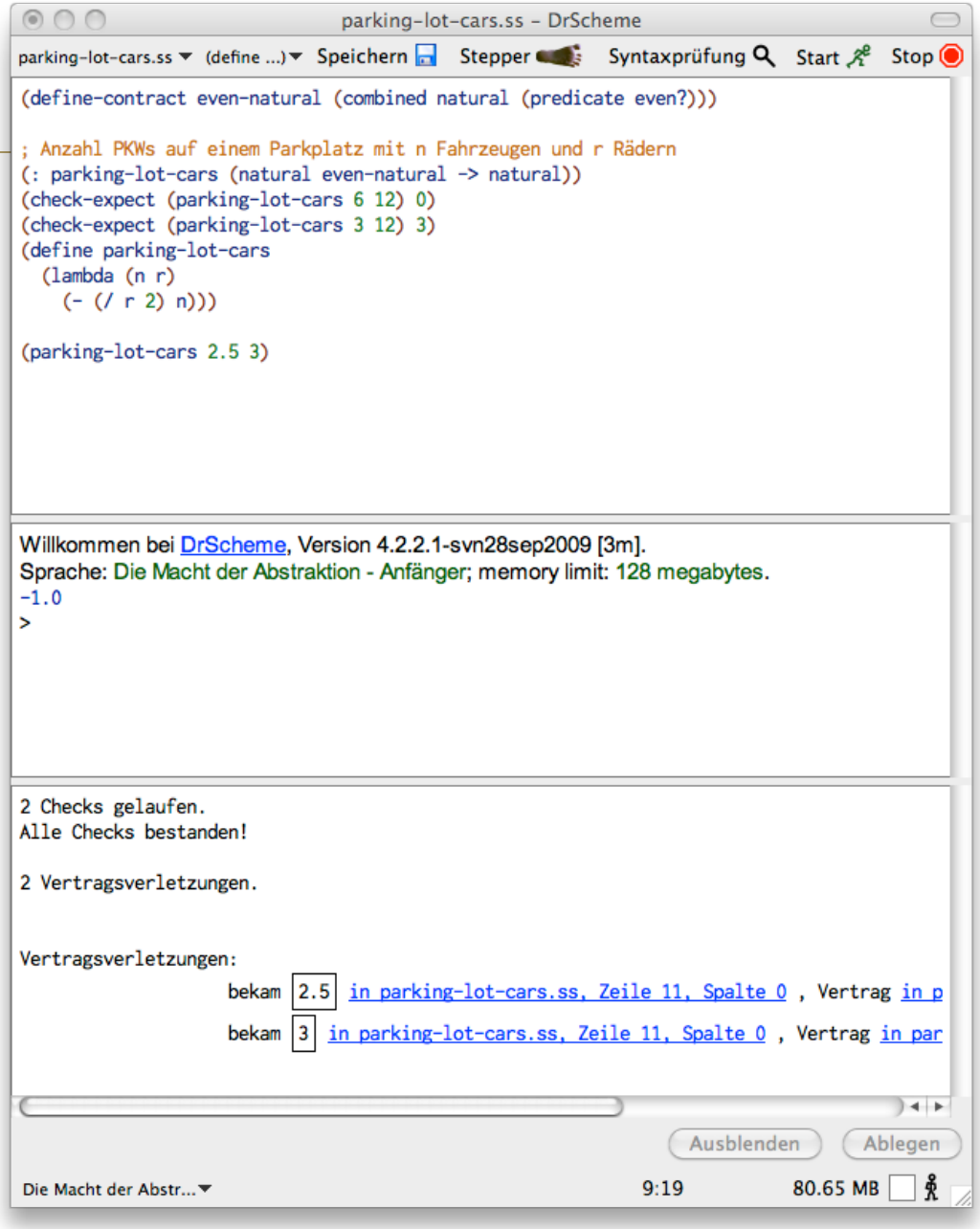
Kein Parkplatz-Problem mehr!

- Problem:
  - Funktion **P(n,r) = .5 (r – 2n)** abstrahiert das Problem und ist allgemeiner anwendbar.
  - Insbesondere auch auf Eingaben, die nicht dem Parkplatzproblem entsprechen können.
    - Ungerade bzw. unmögliche Anzahl von Rädern (P(3,9)).
    - Zu wenige Räder (P(5,2))
    - Zu wenige Fahrzeuge (P(2,10))
- Funktion benötigt einen „Vertrag“.
- Im Beispiel ist das Resultat der Funktion nur korrekt wenn:
  - Anzahl Räder grundsätzlich:
    - r muss gerade sein
  - Nicht zu wenig Räder bzw. zu viele Fahrzeuge:
    - 2n <= r <= 4n
  - Anzahl Fahrzeuge sollte nicht negativ sein: n >= 0

Dr Racket

- Programmierumgebung für **Scheme**, eine Sprache, die funktionales Programmieren (und mehr) unterstützt.

- Interaktiv
- Hilfsbereit
- Konfigurierbar
- Kostenfrei
- Quasi überall verfügbar ☺



Umsetzung in der Funktionalen Programmiersprache Scheme

- Funktionale Programmiersprachen:
  - Funktionen als zentrale Bausteine im Programm
  - Konstruktion von Programmen aus Funktionen
  - Mathematisch: $f(x_1, x_2, \ldots, x_n)$
  - Als funktionales Programm: $(f\ x_1\ x_2\ \ldots\ x_n)$

- In Scheme und anderen funktionalen Programmiersprachen steht der Operator – oder die Funktionsanweisung – vorne (Präfixschreibweise):

  Zum Beispiel:
  - (+ 40 2)
  - (odd? 42)

Vorteile durch Funktionen – Das Leibniz-Prinzip

- Funktionen verhalten sich
  - nachvollziehbar und verlässlich.

- In der Mathematik als eines der Grundprinzipien:
  - Gegeben Funktion f und die Argumente x und y beliebig:

  - Wenn x = y und f(x) = z, dann folgt f(y) = z

x = y und id (x) = z ⇒ id(y) = z

(1646—1716)

- Wesentliche Beiträge zur Mathematik, Paläontologie, Philosophie, Physik, Politik, Rechtswesen, ...

- 1673: Rechenmaschine

Leibnitz-Prinzip und Funktionale Programmierung

- Die weitaus meisten Programmiersprachen sichern den Entwicklern das Leibniz–Prinzip **nicht** zu!
- Somit geht die Vorhersagbarkeit des Verhaltens einer Funktion allein mittels ihrer Argumente verloren.

- *Gegeben* $x = y$ und $f(x) = z \quad \not\Rightarrow \quad f(y) = z$

- Viele intuitive Annahmen gelten bei der Programmkonstruktion dann i.a. nicht mehr.
  - Zum Beispiel: f(x) + f(x) ≠ 2 * f(x)
  - … denn das Verhalten von f kann von einer unübersichtlichen Vielzahl von (impliziten) Parametern abhängen, wie zum Beispiel:
    - vorhergegangene Anwendungen; weiterer Funktionen (inkl. f); Systemzustand; Zeit; Eingaben des Users; Mondphase; ...

Konsequenzen

- Programmkonstruktion unter Einfluss von / mit Wirkung auf den Systemzustand bedarf besonderer Sorgfalt.

- **Funktionale Programmiersprachen** versuchen solche subtilen Einflüsse zu vermeiden…
  - durch Funktionsverträge und
  - durch funktionale Programmierung.

- **Imperative und objektorientierte Programmiersprachen** versuchen solche subtilen Einflüsse auch zu vermeiden…
  - durch Datenkapselungen (private versus public).

Ist Funktionale Programmierung relevant ?

- Durch das Leibniz-Prinzip ist Parallelisierbarkeit grundsätzlich leicht möglich (aber die Details sind immer noch nicht trivial umsetzbar).
  - Gegeben Funktionen f und g, die sich nicht gegenseitig beeinflussen:
  - ➢ Funktionen sind direkt parallel ausführbar.

- Anwendungen:
  - Googles *App Inventor for Android* basiert auf Scheme.
  - Ericsson's AXD301 ATM Switch basiert auf Erlang.
  - Die größten Datenmengen dieser Welt (z.B. Googles Web–Index) werden mit Hilfe von MapReduce — einer "Schablone" zur systematischen Konstruktion von funktionalen, parallelen Programmen—analysiert.
  - Viele Datenbanken werden durch funktionale Programme gesteuert.
  - Compiler werden online durch funktionale Programme auf Konsistenz getestet.

What‘s Next ?

- Erste Schritte mit Dr. Racket und Scheme.