Informatik-Propädeutikum

Dozentin: Dr. Claudia Ermel

Betreuer: Sepp Hartung, André Nichterlein, Clemens Hoffmann

Sekretariat: Christlinde Thielcke (TEL 509b)

TU Berlin
Institut für Softwaretechnik und Theoretische Informatik
Prof. Niedermeier
Fachgruppe Algorithmik und Komplexitätstheorie
http://www.akt.tu-berlin.de

Wintersemester 2013/2014

Gliederung

Organisation

ISIS-Seite für aktuelle Informationen, Übungsmaterial etc.:
https://www.isis.tu-berlin.de/2.0/course/view.php?id=558

- Vorlesung: Montags 12-14 Uhr, H 3010
- (freiwillige! Ergänzung) 1-2 Workshops im dEIn Labor, EMH 028

Prüfungsleistung (im Sinne von Prüfungsäquivalenten Studienleistungen): ein Multiple-Choice-Test und eine schriftliche Endkontrolle (EK). Der Multiple-Choice-Test geht mit 10% in die Endnote ein. Beide Teilleistungen sind kompensierbar.

Literatur: Verschiedene Quellen, z.B.:

- Hal Abelson, Ken Leeden, Harry Lewis: Blown to Bits, 2008.
- Tim Bell, Ian H. Witen, Mike Fellows: Computer Science Unplugged, 1998.
- Jens Gallenbacher: Abenteuer Informatik, 3. Auflage, 2012.
- Christopher Moore, Stephan Mertens: The Nature of Computation, 2011.
- Uwe Schöning: Ideen der Informatik, 3. Auflage, 2008.

Motivation – Worum geht's?

Ziel: Ideengetriebene, leicht verständliche Darstellung von elementaren Grundlagen, Anwendungen und Perspektiven der Informatik.
„Rundumschlag".

Einige zentral behandelte Ideen (= Konzepte, Modelle, Algorithmen, Beschreibungsmethoden, Protokolle, Vorgehensweisen, ...):

- Abstraktion.
- Rekursion.
- Effizienz.
- Information.
- Kodierung und Verschlüsselung.
- Verteiltheit und Nebenläufigkeit.
- Zufall.
- Informatik in Anwendungen.
- ...

Einführung: Was ist ein Propädeutikum?

Siehe Wikipedia:
Das Propädeutikum (von Griechisch propaideuein „im Voraus unterrichten“, von pro „vor“ und paideuein „lehren, unterrichten“) ist meist eine Vorbereitungsveranstaltung auf ein wissenschaftliches Gebiet. Der Begriff wird oft bedeutungsgleich mit Einführungsveranstaltung verwendet; ...

Aus der Modulbeschreibung zur Vorlesung:
... **Einblick in grundlegende wissenschaftliche Fragestellungen und Phänomene der Informatik**[1] ... historische Entwicklung der Informatik ... Kompetenzprofil ... Informatikanwendungsfelder ... Abgrenzung von anderen Wissenschaftsgebieten ... interdisziplinäre Anforderungen an Informatikarbeit ... gesellschaftliche Relevanz... kritische Hinterfragung von Informatikanwendungen und -methoden....

[1] Unser Schwerpunkt hier!

Was ist Informatik?[2]

Wikipedia: Informatik ist die „Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Digitalrechnern“.

Oder: Informatik ist die *Faszination*, sich die Welt der Information und des symbolisierten Wissens zu erschließen und dienstbar zu machen.

Ein Zitat, das oft Edsger W. Dijkstra, dem Wegbereiter der strukturierten Programmierung, zugeschrieben wird, wohl aber Folklore ist:

„In der Informatik geht es genau so wenig um Computer, wie in der Astronomie um Teleskope.“

Edsger W. Dijkstra, 1930–2002

[2] Vergleiche auch das gleichnamige Positionspapier der Gesellschaft für Informatik e.V.

Einordnung in die Wissenschaftslandschaft

Informatik = Information und Automatik (bzw. Information und Mathematik). Im Englischen „Computer Science“; dort wird „Informatics“ eher für bestimmte Teile der angewandten Informatik verwendet (z.B. „Bioinformatics“).

Historisch hat sich die Informatik einerseits als Formalwissenschaft aus der **Mathematik** entwickelt, andererseits als **Ingenieurdisziplin** aus dem praktischen Bedarf nach einer schnellen und insbesondere automatischen Ausführung von Berechnungen.

Informatik ist

- Grundlagenwissenschaft und
- Ingenieurwissenschaft und auch
- Experimentalwissenschaft ($\rightsquigarrow$ Simulationen).

Informatik ist wie die Mathematik eine auf alle anderen Wissensgebiete ausstrahlende Grundlagenwissenschaft; steht bei der Mathematik mehr das „formal Denkbare“ im Vordergrund, so in der Informatik das „Realisierbare“ ($\rightsquigarrow$ konstruktiver Aspekt).

Säulen der Informatik

Klassischer Weise werden die folgenden vier Teilbereiche der Informatik unterschieden, wobei die Grenzen fließend sind:

- Theoretische Informatik;
- Technische Informatik;
- Praktische Informatik;
- Angewandte Informatik.

Die vielfältigen Anwendungsgebiete der Informatik

Informatik besitzt eine ungewöhnliche Breite. Wie die Mathematik ist sie eine Querschnittswissenschaft.

Zusammenwachsende Wissenschaften:

- Bioinformatik
- Geoinformatik
- Medieninformatik
- Medizininformatik
- Wirtschaftsinformatik
- ...

Charakteristisch ist für Informatiker das Arbeiten im **Team** mit Anwendern und Fachleuten **anderer Disziplinen**.

(Kurz-)Geschichtliches

Wichtige Ereignisse für die Informatikgeschichte:

- **Kerbhölzer** als Zähl- und Rechenhilfe vor 30000 Jahren.
- **Abakus** der Chinesen und Römer (seit über 3000 Jahren).
- **Euklid'scher Algorithmus** (ca. 300 vor Chr.): Bestimmung des größten gemeinsamen Teilers zweier ganzer Zahlen. Euklid Autor des einflussreichsten Mathematikbuchs aller Zeiten: „Elemente“ (Geometrie, Zahlentheorie); spezieller Wesenszug des Buchs: konstruktive (!) Existenzbeweise.
- **Sieb des Eratosthenes** (284–202 vor Chr.; geboren in Kyrene (heute Libyen), gestorben in Alexandria): Algorithmus zur Bestimmung aller Primzahlen bis zu einer vorgegebenen Maximalzahl.
- **Heron** von Alexandria (100 nach Christus): Verfahren zum Wurzelziehen (lange zuvor schon bei den Babyloniern...)
- **Chinesischer Restsatz** nach Sun Zi (ca. 200 nach Chr.): $\rightsquigarrow$ Algorithmus zum Lösen modularer Gleichungssysteme.

(Kurz-)Geschichtliches II

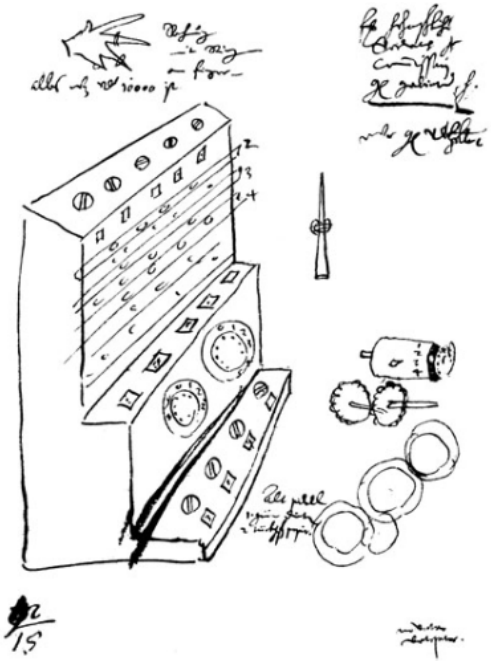
- **Abu Ja'fer Mohammed Ibn Musa Al-Khowarizmi** (ca. 780–850; Persien-Arabien): Verfasser eines sehr einflussreichen **Rechenbuchs**[3] (Dezimalsystem mit Null); abgeleitete lateinische Sprachfloskel „Dixit Algorizmi" („Also Sprach Al-Khowarizmi") als Gütezeichen für die Richtigkeit einer Rechnung $\rightsquigarrow$ Begriff **Algorithmus**.

Al-Khowarizmi auf einer sowjetischen Erinnerungsbriefmarke (1983)

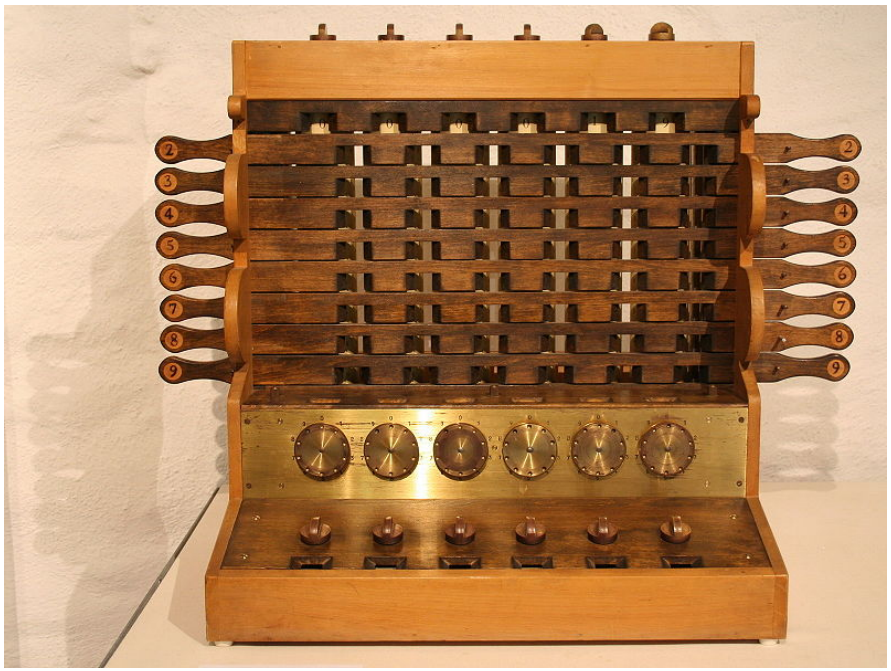
[3] „Rechnen auf der Linie" (1518) von Adam Ries in weiten Teilen eine „Übersetzung" von Al-Khowarizmis Buch.

(Kurz-)Geschichtliches III

Wilhelm Schickard (1592–1635): Rechenmaschine mit Zahnradgetriebe zur Erleichterung astronomischer Rechnungen; Addieren und Subtrahieren von sechsstelligen Zahlen.



Originalzeichnung von Schickard



Nachbau der Rechenmaschine

(Kurz-)Geschichtliches IV

- **Blaise Pascal** (1623–1662): „Pascaline“-Rechenmaschine (für seinen Vater, einen hohen Steuerbeamten) zum Addieren und Subtrahieren. Ca. 50 Exemplare wurden wirklich gebaut.

Eine Pascaline aus dem Jahr 1652

Blaise Pascal

Niklaus Wirth (ETH Zürich) benannte nach ihm die Programmiersprache Pascal, die zur Lehre des strukturierten Programmierens dient(e) (gut prüfbarer und wartbarer Code).

(Kurz-)Geschichtliches V

- Weitere Rechenmaschinen von Gottfried Wilhelm Leibniz (1646–1716), Giovanni Poleni (1683–1761), Antonius Braun (1686–1728), Jacob Leupold (1674–1727), ...

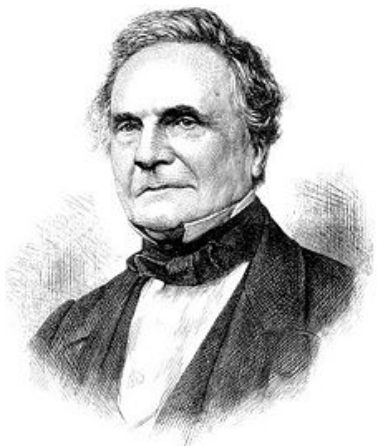
Wesentlich Einschränkung aller obigen Rechenmaschinen:
Von Erbauung an klar festgelegt, was sie tun sollten, sprich festgelegte Funktionalität; keine Trennung von „Hardware“ und „Software“.

Dies änderte sich im 19. Jahrhundert durch die Vision von Charles Babbage: Eine **programmierbare** und damit „universelle“ Rechenmaschine.

Zitat: „As soon as an Analytical Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will then arise — by what course of calculation can these results be arrived at by the machine in the shortest time?“
(Babbage (1864) Passages from the Life of a Philosopher)

(Kurz-)Geschichtliches VI

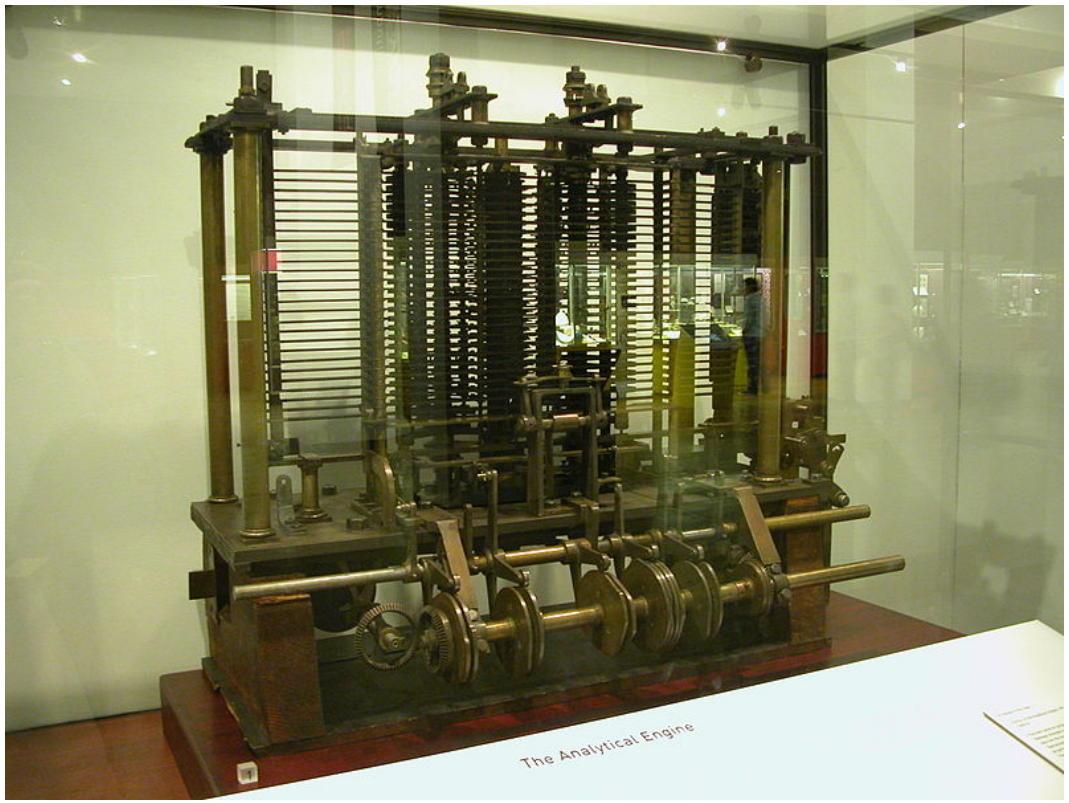
- **Analytical Engine** von **Charles Babbage** (1791–1871) unter Mitarbeit von **Ada Lovelace** (1815–1852) ($\rightsquigarrow$ Namensgebung der Programmiersprache Ada ihr zu Ehren): Als mechanische Rechenmaschine für allgemeine Anwendungen (universell programmierbar!) Vorläufer des modernen Computers; Ada Lovelace beschrieb die Programmierung der Maschine (u.a. Konzept der Schleifen) in der Theorie und gilt daher als erste Programmiererin.



Charles Babbage

Ada Lovelace

(Kurz-)Geschichtliches VII



Versuchsmodell einer Analytical Engine

Geplant: Dampfmaschinenantrieb, Lochkarteneingabe (ähnlich wie bei schon bekannten mechanischen Webstühlen), Speicher für 1000 Wörter mit bis 50 Dezimalstellen (ca. 20,7 kB); vier Grundrechenarten. Programmiersprache ähnlich Assemblersprachen.

(Kurz-)Geschichtliches VIII

- **Konrad Zuse** (1910–1995), Bauingenieur-Alumnus der TU Berlin: Z1 (1938; mechanische Schalter); Z3 aus dem Jahre 1941 erster vollautomatischer, programmgesteuerter und frei programmierbarer Digitalrechner der Welt (basierend auf elektromagnetischer Relaistechnik mit ca. 2200 Relais).

Konrad Zuse

Z3-Nachbau in München

(Kurz-)Geschichtliches IX

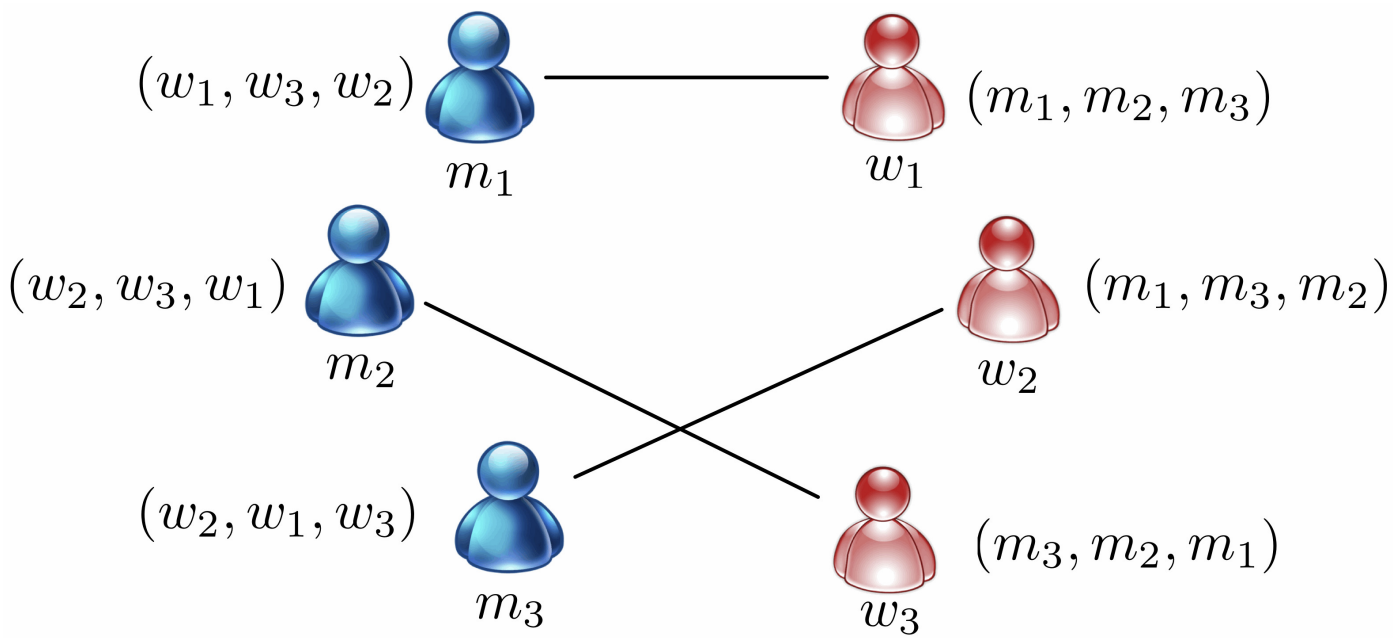
- ENIAC (Electronic Numerical Integrator and Computer): 1946 an der Universität Pennsylvania: Computer auf Basis von Elektronenröhren.
- UNIVAC (Universal Automatic Computer): wirtschaftlich überlebensfähige Weiterentwicklung der ENIAC. Einsatz bei Volkszählung. Aufgrund des Erfolgs stieg nun IBM in das Geschäft mit universellen Rechenmaschinen ein...
- 1956 erster Computer auf Transistorbasis (statt Elektronenröhren)
- ...

Abschließende Bemerkung: Die ersten Programmiersprachen waren sehr maschinennah und wenige Menschen beherrschten die Programmierung.
$\rightsquigarrow$ Programmiererknappheit
$\rightsquigarrow$ Entwicklung von Hochsprachen zur Programmierung: FORTRAN, COBOL, ALGOL (58, 60, 68, W), ...;
$\rightsquigarrow$ Beginn der Dominanz der Software (Algorithmen) gegenüber der Hardware in der Informatik.

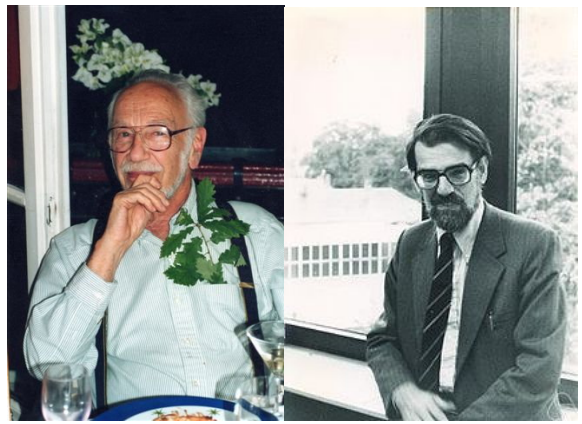
Algorithmen aktuell: Wirtschafts-Nobelpreis 2012

Lloyd Shapley (1923–) gewann 2012 zusammen mit Alvin E. Roth den Wirtschafts-Nobelpreis 2012 für die Theorie stabiler Zuweisungen und den Entwurf praktischer Marktmechanismen.

Ein wesentlicher Aspekt hierbei ist der Gale-Shapley-Algorithmus für „Stable Matching“ (auch bekannt als „Stable Marriage“)...



Stable Matching (Gale und Shapley)



Input: $M = \{m_1, \ldots, m_n\}$ („Männer“)
$W = \{w_1, \ldots, w_n\}$ („Frauen“)
jedes $m \in M$ ordnet alle Elemente aus W nach Präferenz
jedes $w \in W$ ordnet alle Elemente aus M nach Präferenz

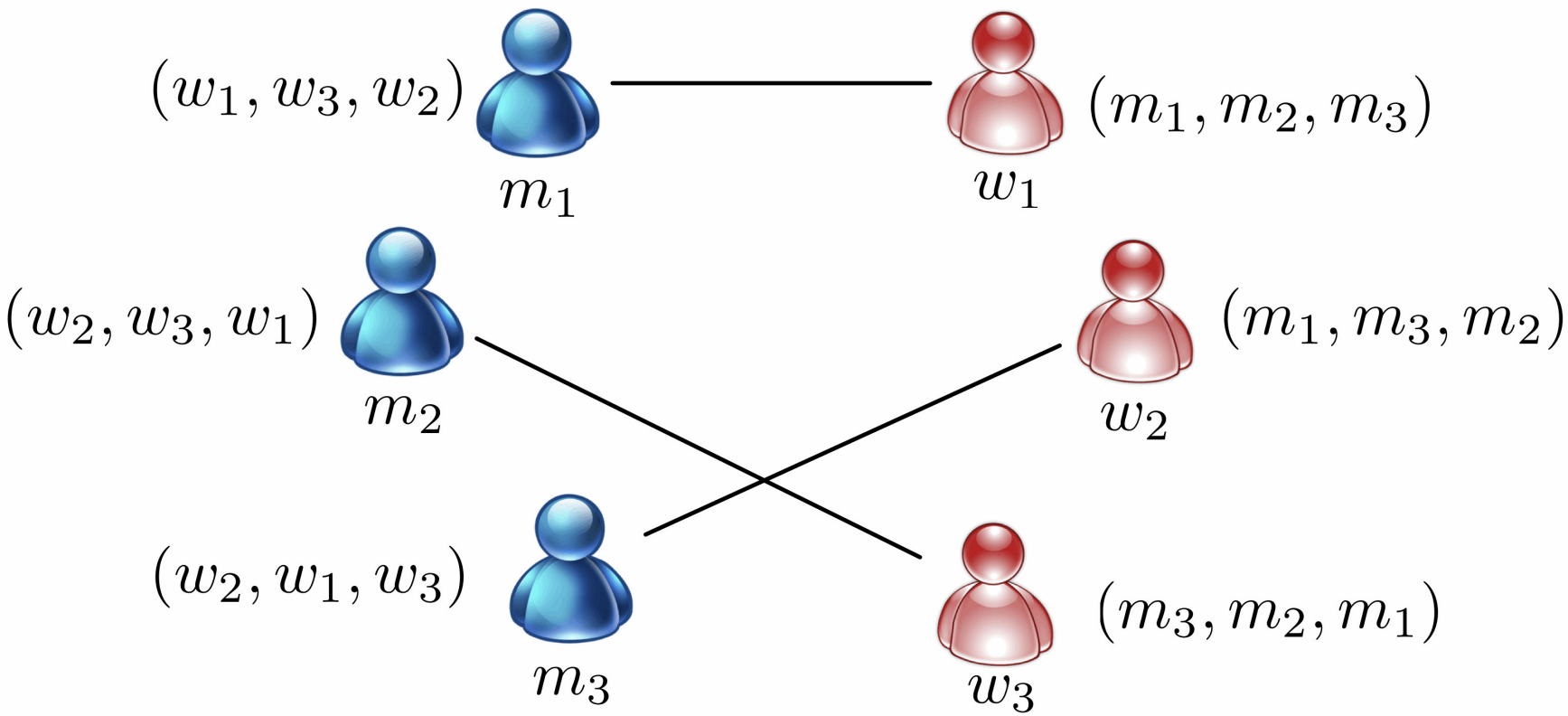
Aufgabe: Finde paarweise Zuordnung zwischen den Elementen aus M und W, so dass für jeden $m \in M$ und jede $w \in W$ die nicht m zugeordnet ist gilt:

1. m zieht ihm zugeordnete w' gegenüber w vor, oder
2. w zieht ihr zugeordneten m' gegenüber m vor.

$\rightsquigarrow$ „**stabile Zuordnung**“

Stable Matching: Beispiel 1

Eine stabile Zuordnung:



Stable Matching: Beispiel 2

Bsp.: $M = \{X, Y, Z\}$, $W = \{A, B, C\}$

$X : A < B < C$	$A : Y < X < Z$
$Y : B < A < C$	$B : X < Y < Z$
$Z : A < B < C$	$C : X < Y < Z$

Test 1: Zuordnung $(X, C), (Y, B), (Z, A)$ stabil?

Test 2: Zuordnung $(X, A), (Y, B), (Z, C)$ stabil?

Effizienter Algorithmus für Stable Matching

Propose&Reject [Gale, Shapley 1962]

1 Initialisiere alle $m \in M$ und alle $w \in W$ als „frei“
2 **while** $\exists w \in W$: w ist frei und $\exists m \in M$, dem w noch keinen Antrag gemacht hat
3 $m \leftarrow$ erster noch „unbeantragter“ Mann in w's Präferenzfolge
4 **if** m ist frei **then:**
5 (m, w) wird Paar, $m \leftarrow$„verlobt“, $w \leftarrow$„verlobt“
6 **else if** m zieht w seiner aktuellen „Verlobten“ w' vor **then:**
7 (m, w) wird Paar, $m \leftarrow$„verlobt“, $w \leftarrow$„verlobt“, $w' \leftarrow$„frei“
8 **else:** m lehnt w ab

Stable Matching: Eigenschaften von Propose&Reject

- Propose&Reject findet immer ein „**perfektes Matching**“ (d.h. bei n Männern und n Frauen werden genau n Paare gebildet), das **stabil** ist, und benötigt dazu $\leq n^2$ **Durchläufe** der while-Schleife.
- Propose&Reject liefert immer eine **„frauenoptimale“** Zuordnung, die zudem für Männer schlechtestmöglich ist.

Fazit: Es lohnt sich, kluge Algorithmen zu entwerfen und sie zu analysieren!

Bemerkungen:

- Viele Problemvarianten: Mehrfachzuordnungen; Unisex; Präferenzlisten ohne totale Ordnung, ...
- (Stable) Matching Probleme haben sehr viele Anwendungen und füllen (zusammen mit ihren Lösungsmethoden) Bücher.