TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

ECON
Institut für Wirtschaftsmathematik
Ökonomie

Ein kurzer Überblick über Netlogo

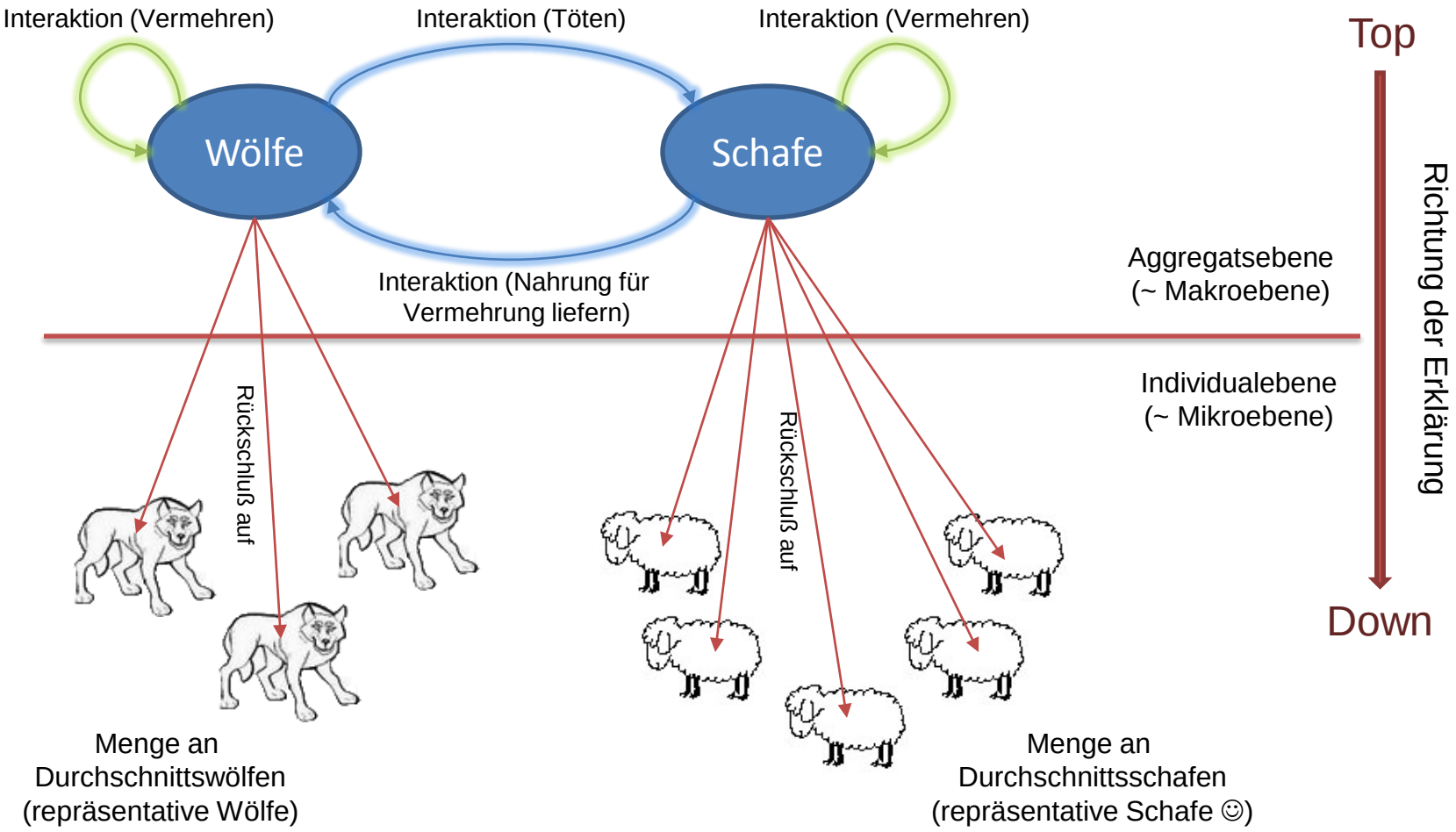
und eine ganz kurze Einführung in ABM

Bernhard Rengs

Natürliche Prozesse

- Natürliche Prozesse (biolog., soziolog.,ökon., etc.)
  - Emergentes Ergebnis des Zusammenspiels/der Interaktion vieler Teile (Agents)
  - oftmals nur durch Aggregate od. Endergebnisse beobachtet
- TOP-Down (bspw. SD) oft ausreichende Erklärung
  - d.h. Rückschlüsse vom Ganzen auf die Teile ziehen
  - bzw. nur TOP: d.h. nur die Gesamtpopulation von Interesse
  - Möglich speziell wenn Individuen sehr homogen
  - Sowie/oder wenn langfristig wenig Veränderung in der Struktur der Interaktion
- aber: TOP Down erscheinen viele Prozesse komplex

TOP-Down Ansatz (SD)



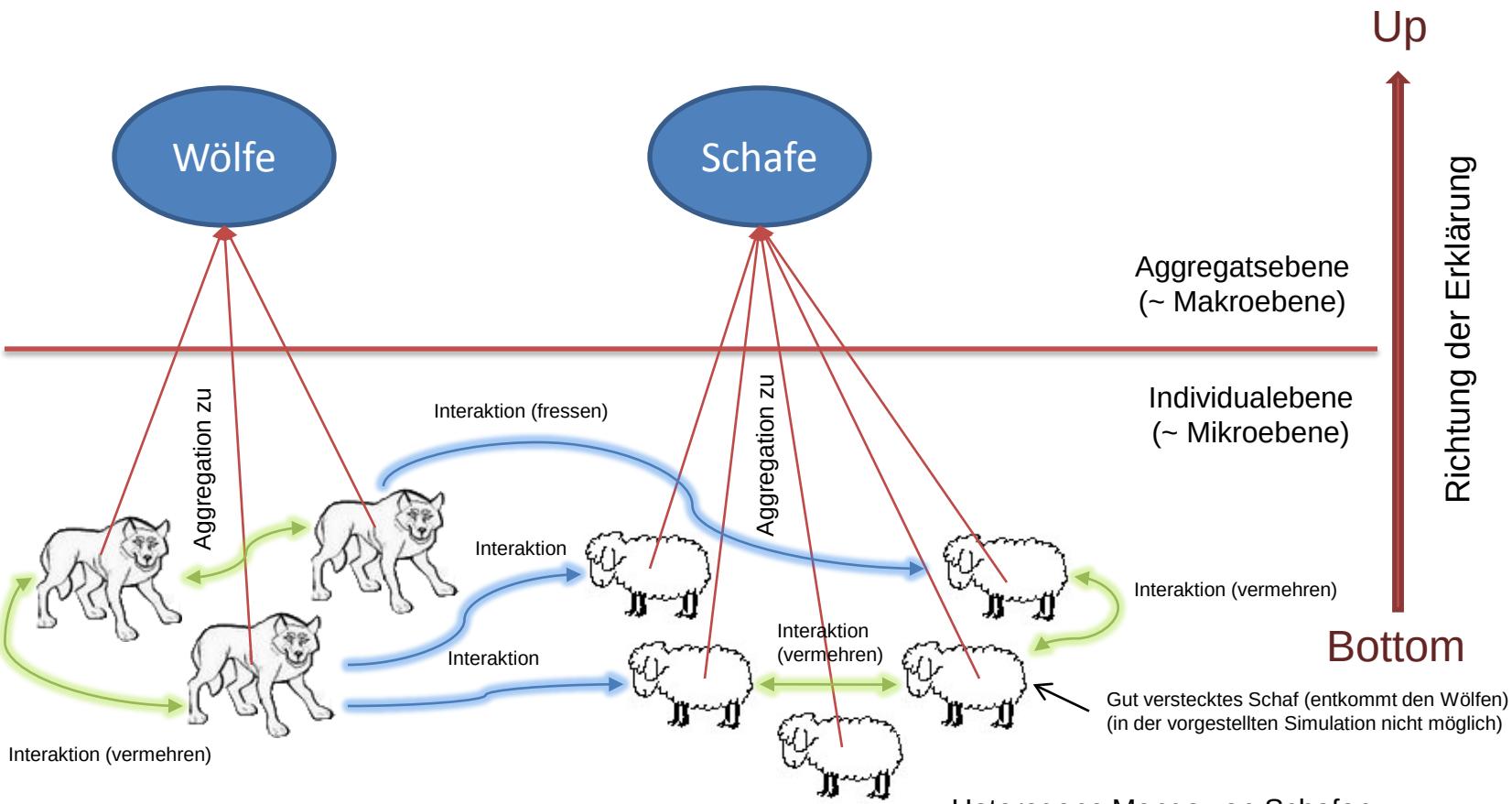
TOP-Down Komplexität

- TD - teilweise unzufriedenstellende Erklärungen
  - Anscheinend unvermittelte Strukturbrüche im Aggregat
    - Ändern sich „Gesetzmäßigkeiten“ im Lauf der Zeit?
  - Verhalten möglicherweise gar chaotisch
    - Keine sinnvolle „Gesetzmäßigkeit“ erkennbar?
- Warum?
  - TOP Down keine akkurate Darstellung dieser Prozesse
  - Änderungen nur auf Ebene der Individuen erkennbar
    - Zusammensetzung d. Population (Demogr.),
    - Räuml. Anordnung (Lokalität),
    - Indiv. Weiterentwicklung (Lernen), usw.

TOP-Down Komplexität (2)

- Außerdem in TD manche Aspekte nicht betrachtbar
  - Lokalität, Heterogene Entwicklung v. Individuen
  - Speziell Soziale Prozesse schwierig: Art d. Vernetzung, Lernen, uvm.
- Einfaches Beispiel: Flocks (flocking.nlogo)
  - Flocks: Wie bewegt sich der Schwarm?
    - Komplexe Bewegungen/Muster des Schwarms
    - Erzeugt durch 3 ganz einfache Regeln der Individuen (Ausrichtung, Distanz bewahren, Nähe Anderer suchen)

Bottom-Up Ansatz



Emergenz im Flocking Beispiel

- Einzelner Vogel hat 3 simple Regeln:
  - Flugrichtung an jene des nähesten Vogels anpassen
  - Mindestabstand halten (Kollisionen vermeiden)
  - Innerhalb der Sichtweite an andere Vögel annähern
- Keine zentrale Planung
  - Keine bestimmter Leitvogel
  - Keine vorgegebene Richtung/Muster
- Dennoch entsteht ein Muster bildender Schwarm
  - Begrenzte Sichtweite, zeitl. Verzögerung bei Anpassungen und Örtlichkeit führen zu nicht geplantem Ganzen
  - Selbstorganisierender Schwarm (selforganizing phenomenon)
  - Scheinbar koordiniertes Vorgehen des Schwarms ist das emergente Phänomen

ABM – agent based modelling

- Form der Erklärung/Darstellung von „Prozessen“
- Bottom-Up Ansatz
  - Beschreibung (Simulation) des Verhaltens (vieler) heterogener Agenten
  - Erklärung des Aggregats als emergentes Ergebnis des Verhaltens der Agentenpopulation
  - Interaktion zwischen Agenten von Interesse (oft auch Struktur dieser)
- Praktisch synonym mit AB-Computational-M
  - Schnelle Berechnung, dadurch möglich: Heterogene Agenten, Große Anzahl Agenten, Dynamische Strukturen, Parameter Sweeps, Direkte Visualisierung, uvvm.

ABM - Ausprägungen

- Unterschiedliche Ausrichtungen/Problemfelder
  - AB – Biology
  - AB – Computational Economics (Theory)
  - AB – Computational Economics (Applied)
  - AB – Computational Sociology
  - AB – Game Theory
  - AB – Geo-Information-Systems (ABM-GIS)
  - uvm.

Netlogo

- Tool/Programmiersprache
  - Grundsätzlich für agent-based spatial modelling gedacht (+ einige Spezialanwendungen)
  - Möglich: SD, CA, ABM, Netzwerke, und Kombinationen
  - Logo Derivat – simple Sprache
  - Netlogo 3D, Hubnet
- Hauptausrichtung
  - Wissensvermittlung (in Form interaktiver Modelle)
  - Kommunikation (Netlogo Prog. immer Open-Source)
  - Prototyping
- Freeware (acedemic use)

Netlogo – Hauptelemente

- Netlogo-Agents
  - Patches (Raum)
    - finite Anzahl, bei Initialisierung festgelegt
  - Turtles (Wesen)
    - dynamisch erzeugbar
    - Breeds möglich
    - shapes veränderbar
  - Links (Verbindung zwischen Turtles)
    - 1:1 Verbindungen, dynamisch erzeugbar,
    - gerichtet & ungerichtet möglich
    - können Attribute haben, shapes veränderbar

Netlogo – Hauptelemente (2)

- Netlogo-Breeds
  - Strukturell unterschiedliche Turtles und Links möglich
  - Unterschiedliche Attribute möglich
    - bspw. „wolves-own [claw-length strength]“
    - bspw. „sheeps-own [wool-color running-speed]
- Agent-Sets (Mengen)
  - Patchset, Turtleset, Linkset
  - Breeds: direkt ansprechbare Teilmengen: „wolves“
  - Teilmengen leicht bildbar „*turtles with [energy > 10]*“
  - bzw. aggregierbar „*(turtle-set wolves sheep)*“

Netlogo – Hauptelemente (3)

- Simulationsstruktur
  - Kein Hauptprogramm/Main Klasse im üblichen Sinn
    - Aufruf über Buttons, bzw. Behaviourspace
  - üblicherweise „setup“ und „go“ Funktion
    - go vs. go forever
- Iteration über Agentsets mit ask
  - Führt gleichen Code hintereinander aus
  - Reihenfolge immer zufällig gezogen (~shuffle list-of-agents)
  - bspw. „*ask patches [with color = red]*“

Netlogo – Vorteile

- Schnelle Entwicklung einfacher Modelle (RP)
- Netlogo selbst in Java (Scala) geschrieben
  - Praktisch auf jedem modernen Computer lauffähig
- Netlogo selbst stetig weiterentwickelt
  - Entwickler reagieren auf Anfragen
  - seit 10 Jahren betreut (aktuell v4.1)
- Breite Benutzerbasis
  - Rel. Umfangreiche Beispiel-Bibliothek
  - Sehr aktive u. hilfsbereite Newsgroup (5-10 Mails / Tag)
    - Häufige Antworten vom Lead-Developer
- (Relativ) Gute Dokumentation

Netlogo – Vorteile (2)

- NL-Befehlszeile während des Laufs verwendbar
- Program kann während des Laufs verändert werden
- Modelle/Programme direkt Web-publizierbar
  - Save-to-Applet Funktion
    - Netlogo-light als Java-Applet
  - Kein Umschreiben, kein Wissen über Java/Applets nötig
  - Nur wenige funktionale Einschränkungen
- Behaviourspace
  - Parameter-Sweep Experimente
  - Replikation

Vorteile – technisch

- Einfache Sprache – Logo Derivat
  - Gut human-lesbarer Code schreibbar:„*ask my-child [move-to one-of patches with [not any? turtles-here]]*“
  - Wenig Programmiererfahrung nötig
  - Relativ geringe Anzahl Elementarer Befehle
  - Vergleichsweise schnelle Simulationen
- Visuelle Komponenten direkt platzierbar
  - Im Code extrem simpel anzusprechen (plotxy, histogram)
- Praktische Dateistruktur (All-in-one)
  - üblicherw. gesamte Simulation in einer Text-Datei
  - inkl. vis. Elemente, Programmcode & Dokumentation

Vorteile – technisch (2)

- Erweiterungen aktiv unterstützt
  - Java Code/Scala kann über Extensions gewrappt werden (d.h. steht dann als Netlogo Befehl zur Verfügung)
  - Es gibt bereits: Arrays, Tables; aber auch GIS Funktionen, Sound, bzw. weitere Mathem. Funktionen
- Netlogoprogramme einbindbar in Java Programme
  - auch Schnittstellen vorhanden: bspw. Mathematica
- Darstellungs & Export Möglichkeiten
  - Movies, Image/Picture Export
  - Export World (Simulation fortsetzbar)
  - CSV-Daten

Nachteile

- Keine Komplettlösung
  - i.d.R. Datenaufbereitung/Analyse mehrerer Läufe extern
- Schwieriger für erfahrene Programmierer (OO)
  - Übliche Programmierkonstrukte fehlen teilweise
    - keine logische Mehrfachverzweigung (case/switch)
    - nur Listen – Arrays und Tables nur per Extension
  - Sprache erzwingt teilw. eigenwillige Strukturen, bspw.:
    - konzentriert auf „entity“ / „agent“ – Gedanken (bspw. „ask“)
    - perverse Art der Wert-Zuweisung: „set var1 var1 + 1“
  - Schwach typisierte Sprache
    - Variablentypen passen sich wenn möglich der Abfrage an – etw. Fehler treten irgendwo auf

Nachteile (2)

- Prozedurale Sprache
  - „Agents“ nur Datencontainer – quasi „recordsets“
    - Sprache nicht (wirklich) objektorientiert
    - Keine Funktionen spezifisch zuordenbar
  - lange Würste teilw. unstrukturierten Codes (Erinnerungen an Endlos-Papier…)
- Schwache Entwicklungsumgebung (IDE)
  - Halbherziges Syntaxhighlighting (breeds-befehle!)
  - Debugging? Was ist das?
    - Nur Syntaxfehler und weniges Semantikfehler während des Codeschreibens erkannt
    - Alle anderen Fehler: Java Exceptions – schwer dem Netlogo-Code zuzuordnen

Nachteile (3)

- Manche Dinge auch über Extensions nicht lösbar
  - Wenige visuelle Elemente & Darstellungsmöglichkeiten
  - Fixe, grauenhafte Farbe von Buttons und Reglern
- Reuse (Weiterverwendbarkeit)
  - Visuelle Elemente nicht einfach in andere Simulation übertragbar
  - Prozeduraler Code schwieriger weiterverwendbar
  - Include-Files machbar, dann aber kein All-in-one mehr
- System Dynamics Modeler
  - nur der Vollständigkeit halber enthalten
  - nur für kleinste Modelle geeignet

Fazit

- Netlogo oft unterschätzt
  - Bei einfachen/mittleren Strukturen/Abläufen
    - (Sehr) Schnelle Programm-Entwicklung
    - Programmieren eher auf Modellfunktionalität konzentriert, weniger auf unterstützende Strukturen
    - Sehr schnell trotz vielen Agenten und langer Laufzeit
  - Leicht kommunizierbar, austauschbar, verständlich
- Aber dennoch Einschränkungen
  - Bei mittleren/sehr langen (Komplexität)
    - hohe Programmierdisziplin gefordert (Selbstdiziplin)
    - Schlechte IDE, Schweres Debugging
  - Im Vergleich zu OO-Sprachen Code schlecht zerteilbar (Reuse, Collaboration)