Logische Programmierung und Constraint Programmierung als Studieninhalte für BWL-Studiengänge - *Motivation, Vorgehen und Erfahrungen*

Ulrich John

Hochschule für Wirtschaft, Technik und Kultur (HWTK)
ulrich.john@hwtk.de

Abstract: Ein belastbares Überblickswissen über die Logische Programmierung und die Constraint-(Logische) Programmierung ist in der heutigen Zeit nach Erfahrung und Meinung des Autors auch für (angehende) Betriebswirte und Wirtschaftsinformatiker wichtig beziehungsweise dringend empfehlenswert, wobei insbesondere Aspekte der deklarativen Programmierung und deren Anwendbarkeit für komplexe betriebswirtschaftliche Problemstellungen eine besondere Rolle spielen. Gerade auch im Kontext der Digitalisierung/ Digitalen Transformation können beide Programmierparadigmen nutzbringend verwendet werden. Im Paper werden einige ausbaufähige Lehrinhalte mit Aspekten zu erprobten Vorgehensweisen vorgestellt und Erweiterungsmöglichkeiten benannt.

Keywords: Constraint Programmierung, Logische Programmierung, Künstliche Intelligenz, BWL- & Wirtschaftsinformatik-Studium, Wissensmanagement, Entscheidungsunterstützungssysteme, Digitalisierung, Digitale Transformation, komplexe Planungsprobleme

1 Einleitung

Studierende der Betriebswirtschaftslehre und der Wirtschaftsinformatik werden im Laufe ihres Berufslebens mit hoher Wahrscheinlichkeit mit Aufgabenstellungen der Digitalisierung und/ oder Digitalen Transformation sowie den damit verbundenen Herausforderungen konfrontiert werden oder werden mindestens im Kontext dieser tätig sein. Obwohl die Digitalisierung von Wirtschaft und Gesellschaft schon seit mehreren Jahrzehnten voranschreitet und entsprechende – teilweise drastische - Transformationen vonstattengingen, ist sie mittlerweile – unter anderem aufgrund der Reife von IT-Technologien und anhaltendem Wettbewerbsdruck - zu einem nahezu „allbestimmenden Themenkomplex“ geworden. Dieser Tatsache muss auch bei der Ausgestaltung von BWL-Studieninhalten sowie bei der Beratung bereits tätiger Manager mit besonderem Augenmerk Rechnung getragen werden. Neben anderen Technologien der Künstlichen Intelligenz spielen aus unserer Sicht – in zunehmendem Maße - die Logische Programmierung und die Constraint Programmierung/ Constraint Logische Programmierung eine entscheidende Rolle, so dass auch diese entsprechend thematisiert werden sollten. Aus der Sicht des Autors kann dies mit Bezug

und in Ergänzung zu Lehrinhalten des Operations Research erfolgen (siehe z.B. [Do15]). Die geeignete Aufnahme der Logischen Programmierung und der Constraint Programmierung/ Constraint Logischen Programmierung in die BWL-Lehre stellt eine besondere Herausforderung dar, da diese beiden Paradigmen ja selbst in verschiedenen Informatik-Studiengängen nicht oder nur rudimentär „Einzug gefunden haben", wobei dieses aufgrund intendierter Spezialisierungen durchaus eine Berechtigung haben kann. Unverständlich ist es unserer Meinung nach jedoch bei Wirtschaftsinformatik-Lehrbüchern. In Standardlehrwerken der Wirtschaftsinformatik (z. B. [Me17], [La16], [La18]) werden zum Beispiel zwar Ausführungen zur objektorientierten Programmierung geboten, Informationen zur Logischen Programmierung und Constraint Programmierung sucht man jedoch derzeit vergeblich.
In den folgenden Abschnitten werden nach einer kurzen Motivationsuntermauerung einige geeignete Beispiele – ohne Anspruch auf Vollständigkeit - für die Inhalts- und Vorgehensgestaltung der LP[1]/ C(L)P[2]-bezogenen Lehrkomponenten von Informatik-/ Wirtschaftsinformatik-Kursen für Studierende der Betriebswirtschaftslehre präsentiert, die an der Hochschule für Wirtschaft, Technik und Kultur seit 5 Jahren erfolgreich verwendet werden[3]. Basierend darauf lassen sich – je nach Zeitrahmen - weitere (betriebswirtschaftliche) Problemstellungen beziehungsweise Lösungsansätze für diese aufbauen, so dass sich mit Anreicherung um Theorieaspekte auch Konzepte für gesamte LP- oder/ und CP-Kurse im Rahmen von Wirtschaftsinformatikstudiengängen o.ä. aufbauen lassen, wie dies zum Beispiel für den HWTK-Studiengang „Informatik und Management" praktiziert wird.

2 Motive

Die Rollenbilder von Betriebswirten haben sich in den vergangenen Jahren deutlich gewandelt und werden sich in Zukunft teilweise drastisch weiter wandeln. Viele menschliche Arbeitsinhalte werden durch Softwarefunktionalitäten substituiert werden (Digitalisierung, Digitale Transformation, vgl. [Jo15], [Jo17]). Dies wird nicht nur „einfache" Verwaltungs- und Beratungsfunktionen betreffen, sondern auch komplexe Managementprozesse, für die es mindestens leistungsstarke Entscheidungsunterstützungssysteme geben wird. Dies wird insbesondere durch die Reifung von IT-Technologien und durch disruptive Technologien ermöglicht, wobei Technologien der Künstlichen Intelligenz eine besondere Rolle spielen werden. Der Erfahrung nach waren in den vergangenen Jahren Studierende der Betriebswirtschaftslehre zu einem erheblichen Anteil „wenig Informatik-affin". Die meisten von Ihnen verfügen zum Beispiel über keinerlei Kenntnisse auf den Gebieten Programmierung und Softwareentwicklung. Eine ablehnende oder ignorierende Haltung gegenüber IT-Inhalten und

[1] Logische Programmierung
[2] Constraint (Logische) Programmierung
[3] davor in Teilen bereits im Bachelor-Studiengang Betriebswirtschaftslehre der Hochschule Lausitz (2011) und im Masterkurs „Algorithms and Optimizations" (2009 – 2011) des Studienganges „Internationale Medieninformatik" der HTW Berlin (in anderem inhaltlichen Kontext und mit anderer didaktischer Einbindung).

IT-gestützten Prozessen wird perspektivisch nicht mehr praktikabel sein, da eine erfolgreiche Betätigung im Berufsleben in aller Regel nur mit entsprechenden Kenntnissen der Informatik/ Wirtschaftsinformatik möglich sein wird. Selbstverständlich muss und soll ein Betriebswirt nicht ein Informatiker werden/ sein, Ausrichtung und Detailierung der IT-spezifischen Kenntnisse müssen auf einem „geeigneten Niveau" vorliegen. Insbesondere sollte das Wissen aktuelle Kenntnisse bezüglich konzeptioneller und technologischer Möglichkeiten und bezüglich der damit verbundenen Herausforderungen, z. B. Kosten, Entwicklungsdauer und Risiken, umfassen.

3 Mögliche Inhalte & erprobtes Vorgehen

Die LP- und CP/CLP-thematisierenden Lehrkomponenten sind in die Wirtschaftsinformatikkurse thematisch einzubetten und ggf. in inhaltlichen Kontext zum Operations Research zu stellen. Analysiert man die Wirtschaftsinformatik-Curricula von BLW-Studiengängen verschiedener Hochschulen/ Universitäten, so sind deutliche Unterschiede festzustellen. An der HWTK wird neben den klassischen Inhalten insbesondere Wert auf der Vermittlung von Überblickwissen über aktuelle Technologien, Paradigmen und IT-Themenfelder (z.B. Cloud Computing, SOA, BIG Data, IT-Sicherheit & Datenschutz, Digitale Transformation, Industrie 4.0 usw.) gelegt, so dass die Studierenden diesbezüglich ein belastbares und zukunftssicheres Wissen erwerben. Ein separater, zeitlich nachgelagerter, Teilkurs widmet sich derzeit dem Themengebiet Datenbanken.
Umfang und inhaltliche Tiefe der Lehrkomponenten können je nach zur Verfügung stehender Zeit variiert werden. Wir veranschlagen für die Themenkomplexe LP & CP derzeit in der Regel bis zu 4 Unterrichtseinheiten a 45 Minuten.

3.1 Thematische Vorarbeit

Die LP- und CP/CLP-Lehrkomponenten werden in den Themenkomplex Programmierparadigmen eingebettet. Hier stellt sich die Frage, ob die deklarative Programmierung schwerpunktmäßig vor der imperativen Programmierung behandelt werden sollte oder danach, insbesondere da die meisten BWL-Studierenden über keinerlei Programmierkenntnisse verfügen. Bei uns wurden in verschiedenen Semestern beide Reihenfolgevarianten praktiziert, wobei eine Präferenz herausgebildet wurde, die deklarative Programmierung vor der imperativen Programmierung zu behandeln. Zeitlich vorgelagert werden die Studierenden unter anderem mit den Begrifflichkeiten *Berechenbarkeit, Entscheidbarkeit, kombinatorische Komplexität*[4] und mit exemplarischen Problemklassen aus dem Alltag von Unternehmen vertraut gemacht. Untermauert werden kann dies mit „Stories aus der Praxis", da viele Betriebswirte/ Manager diesbezüglich kein belastbares Wissen besitzen und daher leider Fehleinschätzungen und Managementfehler entstehen, die ggf. zum Scheitern von Projekten führen und/ oder erhebliche, vermeidbare Kosten verursachen.

[4] zum Beispiel anhand des Beispiels der Türme von Hanoi

3.2 Logische Programmierung

Beim Themenkomplex *Logische Programmierung* beginnen wir mit der Vorstellung des grundlegenden Prinzips des Paradigmas der Logischen Programmierung und der damit verbundenen Begrifflichkeiten (Wissensbasis, Fakten, Regeln, Inferenzmaschine, Anfragen, Resultate), ohne ins Detail zu gehen.

Anschließend präsentieren wir verschiedene grundlegende hierarchische Strukturen aus betriebswirtschaftlichem Kontext, z.B. Zuliefernetze, Organigramme, Produkt- und Prozessbäume. Anhand dieser diskutieren wir praktische Fragestellungen, die unter anderem das Themenfeld Informations- und Wissensmanagement betreffen. Anhand eines Beispielproblems präsentieren wir (bei ausreichender Zeit ist eine gemeinsame Entwicklung didaktisch sinnvoll) ein Prologprogramm, dass dieses Problem in Form von Regeln und Fakten spezifiziert und erklären dieses. Für das „Top-Boss-Problem“ (Unternehmensorganigramm) kann das folgendermaßen aussehen:

```
vorgesetzter(X, Y):-
        direkterVorgesetzter(X, Y).
vorgesetzter(X, Y):-
        direkterVorgesetzter(X, Z),
        vorgesetzter(Z, Y).
topBoss(X, Y):-
        vorgesetzter(X, Y),
        not(direkterVorgesetzter(_, X)), !.     % ! ist nichtdeklarative Ausnahme
direkterVorgesetzter(heinzMeier, erwinMüller).
direkterVorgesetzter(drDieterZetsche, heinzMeier).
…
```

Anhand von exemplarischen Anfragen wird die Vorgehensweise der Inferenzmaschine einschließlich *Backtracking* (grob) erklärt; Beispielanfrage:

? topBoss(X, karlWichtigtuer).

Ausgabe: *X = drDieterZetsche*

Bei zeitlicher Passung könnte man nun eine Erweiterung der präsentierten Problemstellungen vornehmen, die komplexe Management-Fragestellungen adressieren und die dann Übungsinhalte für die Studierenden bilden. Beispiele hierfür sind unter anderem die Aufnahme von zeitlichen Aspekten/ Historieninformationen in Liefernetze oder in Organigramme. Zusätzlich (mit Hinweis auf Analogien) oder alternativ könnte nicht-betriebswirtschaftliches „Alltagswissen“ modelliert werden, z.B. wie klassisch oft getan, Wissen über Verwandtschaftsbeziehungen.
Insgesamt sollte den Studierenden klar werden, welche Potenziale die Logische Programmierung für die Realisierung von effizientem Informations- und Wissensmanagement besitzt. Eine Inkontextsetzung zu weiteren Aspekten/ Inhalten des Wissensmanagements sollte an geeigneter Stelle erfolgen.

3.3 Constraint Programmierung/ Constraint Logische Programmierung

Je nach zur Verfügung stehender Zeit kann man (klassisch) mit der Modellierung eines „Buchstabenrätsels“, wie zum Beispiel SEND + MORE = MONEY (vgl. z.B. [AW07]) starten, die den Studierenden ein Gefühl vermitteln, wie Problemstellungen deklarativ modelliert werden können. Eine Lösungsfindung sollte einerseits durch reine *logische Programmierung* und andererseits durch *constraint-logische Programmierung* realisiert werden. Beide Ansätze sollen hinsichtlich der Laufzeit verglichen werden.
Wir wählen in der Regel initial das „Problemfeld“ der *magischen Quadrate*, wobei sich durch zusätzliche Informationen, etwa aus [He09], das Interesse der Studierenden steigern lässt. Wir starten mit magischen Quadraten in der 3x3-Ausprägung.
Es bietet sich an, ein solches zunächst durch ein rein logisches Programm zu spezifizieren, wobei die Studierenden die einzuordnenden aufeinander folgenden Zahlen (hier 25 bis 33) selbst wählen dürfen, z.B.:

```
start:-
    cputime(Ts),
    % jede Zahl soll genau einmal vertreten sein
    member(25,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(26,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(27,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(28,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(29,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(30,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(31,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(32,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    member(33,[A1,A2,A3,A4,A5,A6,A7,A8,A9]),
    % gleiche Zeilensumme
    Z1 is A1 + A2 + A3,
    Z2 is A4 + A5 + A6,
    Z3 is A7 + A8 + A9,
    Z1 == Z2, Z2 == Z3,
    % gleiche Spaltensummen
    S1 is A1 + A4 + A7,
    S2 is A2 + A5 + A8,
    S3 is A3 + A6 + A9,
    Z3 == S1, S1 == S2, S2 == S3,
    % gleiche Diagonalensummen
    D1 is A1 + A5 + A9,
    D2 is A3 + A5 + A7,
    S3 == D1, D1 == D2,
    cputime(Te), T is Te - Ts,
    write('Loesung: '),nl,
    write([A1,A2,A3]),nl,
    write([A4,A5,A6]),nl,
```

```
    write([A7,A8,A9]),nl,
    write('gefunden in '),write(T), write(' msec.'),nl.
```

Nach Aufruf der Prädikates *start* erfolgt relativ schnell die Ausgabe einer Lösung, z.B. mit CHIP unter Windows 7 auf einem i5-3230M, 2,6 GHz mit 12 GB Hauptspeicher in 234 msec (CPU-Zeit):
[26, 31, 30]
[33, 29, 25]
[28, 27, 32]
gefunden in 234 msec.

Ein analoges (vorbereitetes) Programm für die Berechnung eines 5x5-Magischen Quadrates könnte nun „gestartet“ werden. Da die Lösungsfindung erhebliche Zeit benötigt, könnte man nun entweder erste einfache betriebswirtschaftliche kombinatorische Planungsprobleme diskutieren und modellieren oder aber die prinzipielle Funktionsweise der Constraint (Logischen) Programmierung erläutern. Alternativ, didaktisch unseres Erachtens besser platziert, wird die Funktionsweise nach den „praktischen Übungen“ und den damit verbundenen Erfahrungssammlungen vorgestellt.
Eine Lösung für die Berechnung des 5x5-Magischen Quadrates lässt dann „immer noch auf sich warten“, daher sollte nun die Modellierung des 3x3-Problems *constraint-logisch* erfolgen[5], wobei durch Zuweisung der „Matrix“ zu einer Variable und deren anschließende Verwendung den Programmcode verkürzen, vermutlich aber die „erste Verständlichkeit“ auch erschweren würde:

```
start:-
    [A11,A12,A13,
    A21,A22,A23,
    A31,A32,A33]::25..33,
    S::0..10000,
    S #= A11 + A12 + A13,
    S #= A21 + A22 + A23,
    S #= A31 + A32 + A33,
    S #= A11 + A21 + A31,
    S #= A12 + A22 + A32,
    S #= A13 + A23 + A33,
    % Diagonalen
    S #= A11 + A22 + A33,
    S #= A31 + A22 + A13,
    alldifferent([A11,A12,A13,A21,A22,A23,A31,A32,A33]),
    suchewerte([A11,A12,A13,A21,A22,A23,A31,A32,A33]),
    schreib9([A11,A12,A13,A21,A22,A23,A31,A32,A33]).
```

[5] hier mittels *CHIP* (Constraint Handling in Prolog, www.cosytec.com) dargestellt, alternativ lassen sich unter anderem *ECLiPSe* (www.eclipseclp.org) oder *GNU Prolog* (www.gprolog.org) einsetzen.

```
suchewerte([]).
suchewerte([H|T]):-
    indomain(H),
    suchewerte(T).

schreib9([A11,A12,A13,A21,A22,A23,A31,A32,A33]):-
    write(A11),write('  '),write(A12),write('  '),write(A13),nl,
    write(A21),write('  '),write(A22),write('  '),write(A23),nl,
    write(A31),write('  '),write(A32),write('  '),write(A33),nl.
```

Die unmittelbare Lösungslieferung,

```
26  31  30
33  29  25
28  27  32  (in 15 msec gefunden)
```

sollte bei den Studierenden die Effekte der Constraint Programmierung erkennbar machen. Für die Lösungsfindung wurden 15 msec statt 234 msec benötigt. Man kann die Aufgabenstellung nun erweitern und die Berechnung aller Lösungen verlangen, was durch gezieltes „failen" und dem dadurch ausgelösten Backtracking realisiert und demonstriert werden kann. Nach dieser Demonstration und der Erkenntnis, dass das „reine" Prologprogramm zur Berechnung eines 5x5-Magischen Quadrates immer noch läuft[6], gilt es, die Euphorie hinsichtlich des CP-Ansatzes zu begrenzen und sich dem Thema „*redundante Constraints*" zu nähern. Dafür starten wir nun ein CP-Programm[7] für 5x5-Quadrate (gleiche Zahlen wie im Prologprogramm) [8]:

```
start:-
    [A11, A12, A13, A14, A15,
    A21, A22, A23, A24, A25,
    A31, A32, A33, A34, A35,
    A41, A42, A43, A44, A45,
    A51, A52, A53, A54, A55]::423..447,
    S::0..10000, % Summe
    % Zeilen-Constraints
    S #= A11 + A12 + A13 + A14 + A15,
    S #= A21 + A22 + A23 + A24 + A25,
    S #= A31 + A32 + A33 + A34 + A35,
    S #= A41 + A42 + A43 + A44 + A45,
    S #= A51 + A52 + A53 + A54 + A55,
```

[6] Man kann sich darauf einstellen, dass auch zum Ende der Lehrveranstaltung weder eine Lösung berechnet wurde noch eine Nichtlösbarkeit nachgewiesen wurde.

[7] Das präsentierte Programm ist ein ad-hoc-Programm und könnte natürlich kompakter und „eleganter" formuliert werden.

[8] zur Auflockerung kann man an der Tafel/ am Whiteboard „Zahlenkünstler-Tricks" präsentieren, mit denen sich magische Quadrate mit ungerader Kantenlänge lösen lassen. Es sollte darauf hingewiesen werden, dass es für Magische Quadrate mit ungerader Kantenlänge ein konstruktives Berechnungsverfahren gibt und dass es solche Verfahren für kombinatorisch komplexe Probleme praktischer Natur i.d.R. nicht gibt.

```
    % Spalten-Constraints
    S #= A11 + A21 + A31 + A41 + A51,
    S #= A12 + A22 + A32 + A42 + A52,
    S #= A13 + A23 + A33 + A43 + A53,
    S #= A14 + A24 + A34 + A44 + A54,
    S #= A15 + A25 + A35 + A45 + A55,
    % Diagonal-Constraints
    S #= A11 + A22 + A33 + A44 + A55,
    S #= A51 + A42 + A33 + A24 + A15,
    alldifferent( [A11, A12, A13, A14, A15,
              A21, A22, A23, A24, A25,
                A31, A32, A33, A34, A35,
                A41, A42, A43, A44, A45,
                A51, A52, A53, A54, A55]),
    cputime(T1),
    suchewerte( [A11, A12, A13, A14, A15,
        A21, A22, A23, A24, A25,
          A31, A32, A33, A34, A35,
          A41, A42, A43, A44, A45,
          A51, A52, A53, A54, A55]),
    cputime(T2),
    schreib_25([A11, A12, A13, A14, A15,
        A21, A22, A23, A24, A25,
          A31, A32, A33, A34, A35,
          A41, A42, A43, A44, A45,
          A51, A52, A53, A54, A55]),
        Tg is T2 - T1,
        write('geloest in '), write(Tg), write(' msec').

schreib_25([A11, A12, A13, A14, A15,
        A21, A22, A23, A24, A25,
          A31, A32, A33, A34, A35,
          A41, A42, A43, A44, A45,
          A51, A52, A53, A54, A55]):-
    write(A11),write('  '), write(A12), write('  '),write(A13),write('  '),write(A14),write('  '),write(A15),nl,nl,
    write(A21),write('  '), write(A22), write('  '),write(A23),write('  '),write(A24),write('  '),write(A25),nl,nl,
    write(A31),write('  '), write(A32), write('  '),write(A33),write('  '),write(A34),write('  '),write(A35),nl,nl,
    write(A41),write('  '), write(A42), write('  '),write(A43),write('  '),write(A44),write('  '),write(A45),nl,nl,
    write(A51),write('  '), write(A52), write('  '),write(A53),write('  '),write(A54),write('
'),write(A55),nl,nl,nl.
```

Die Lösungsfindung benötigt trotz CLP-basierter Modellierung erhebliche Zeit. Die Studierenden sollen nun überlegen, welches Wissen man über die Lösung oder die Struktur der Lösung noch generieren kann, ohne eine Lösung zu kennen. In der Regel kommt jemand darauf, dass man die Summe einer Zeile/ einer Spalte/ einer Diagona-

len berechnen kann, wenn man die einzuordnenden Zahlen kennt (mit unseren gewählten Werten ist die Summe 2175; es bittet sich an, in diesem Kontext die von Gauß gefundene Berechnungsmethode zu wiederholen). Wenn nicht, muss man diese Idee selbst herleiten. Diese Erkenntnis nutzend wird zusätzlich das redundante Constraint *S #= 2175* eingefügt. Ein erneuter Start der Problemlösung liefert rasch eine Lösung:

423 424 435 446 447
425 444 441 428 437
445 438 432 433 427
443 429 431 442 430
439 440 436 426 434
geloest in 125 msec

In diesem Kontext wird über die Bedeutung von redundanten Constraints diskutiert. Als didaktisch sinnvolle Programmierübung wird das Programm nun flexibilisiert, d.h., die Berechnung des Summenwertes wird durch das Programm durchgeführt, somit können durch das Programm ohne weiteres verschiedene Probleminstanzen der Problemklasse *5x5-Magische Quadrate* gelöst werden.

Nun widmen wir uns in der Regel einer nächsten, komplexeren Problemklasse (7x7-Quadrate). Hierbei wird klar, dass das Hinzufügen der Zielen-/ Spalten-/ Diagonalensumme als redundantes Constraint nicht ausreichend ist, um in kurzer Zeit eine Lösung zu finden. Bei ausreichend zur Verfügung stehender Zeit oder aber als Inhalt „studentischer Überlegungen“ in Begleitung zur Lehrveranstaltungsreihe kann man hier weiterführende Ideen „beauftragen“ und testen lassen. Es bietet sich an, hier das Thema *Heuristiken* zu thematisieren.

Wenn nicht schon geschehen (vgl. oben) sollte nun das allgemeine Prinzip der Constraint-Programmierung (einschließlich Propagation[9]) vorgestellt werden. Praktische Problemklassen aus der Betriebswirtschaft, die durch Constraint-Programmierung adressiert werden können, sollten wiederholt/ vorgestellt und diskutiert werden, z.B. *Ablaufplanungsprobleme*, *Fließproduktionsplanung*, *Konfigurationsprobleme* und Multiressourcen-Planungsprobleme, ggf. mit Bezug zum Operations Reasearch, wobei auch hier die Bedeutung der *Deklarativität* der jeweiligen Problemspezifikation thematisiert werden sollte.

Weitere Vorteile (*Unterstützung von flexibler Interaktivität*, *Anwendbarkeit trotz unvollständigen Wissens* usw.), Potenziale und für praktische Problemlösungen ggf. erforderliche Erweiterungen (z.B. *weiche Constraints*, *Constraint-Hierarchien*, *Präferenzen*) sowie weiter bestehende Herausforderungen der Constraint Programmierung sollten deutlich und einprägsam benannt werden. Erwähnt werden sollten (ohne detaillierte Behandlung) zum Beispiel insbesondere auch *globale Constraints*, z. B. anhand des in den Beispielen verwendeten *alldifferent-Constraints* und deren Bedeutung mit Bezug zum Operations Research. Es sollte darüber hinaus darauf hingewiesen werden, dass für praktische Problemklassen unter Umständen die Notwendigkeit von hybriden Ansätzen/ Modellen erforderlich ist.

[9] zum Beispiel anhand der Intervallgrenzenpropagation

Um die inhaltliche Verfestigung und Wiederholung zum Beginn einer nächsten Wirtschaftsinformatik-Lehrveranstaltung zu unterstützen, rufen wir in der Regel dazu auf, „richtig schwere“ Sudokus „zu ermitteln“. Diese werden dann im Kontext einer inhaltlichen Wiederholung constraint-basiert gelöst. Für ergänzende Informationen sei hierfür auch auf [Si05] verwiesen.

4 Erfahrungen

Wie in Kapitel 2 erwähnt, sind zum Beginn des BWL-Studiums „relativ viele“ Studierende „wenig Informatik-affin“. Der Erfahrung des Autors nach wird bei einem erheblichen Anteil dieser Studierenden durch Integration der beschriebenen Lehrinhalte in der benannten Vorgehensweise ein „gewisses Interesse“ und Verständnis für Informatik/ Wirtschaftsinformatik geweckt/ ermöglicht. Die Studierenden akzeptieren, dass die Beschäftigung mit „aktuellen IT-Themen“ i.d.R. unabdingbar und nutzbringend ist, wenn man ein zukunftssicheres Studium absolvieren will.
Die Tatsache, dass die deklarative Programmierung zeitlich vor der imperativen Programmierung eingeführt wird, begünstigt unseres Erachtens die studentische Erkenntnis, dass Programmieren relativ einfach sein kann und erhöht die Bereitschaft, sich auch mit anderen Themengebieten der Informatik/ Wirtschaftsinformatik zu beschäftigen. Durch die Inkontextsetzung zu betriebswirtschaftlichen Problemstellungen und Managementaufgaben sowie durch die Einbettung in das „big picture“ *Digitales Unternehmen* (vgl. [Joh15], [Joh17]) erkennen die Studierenden die Bedeutung deklarativer Spezifikationen und der deklarativen Programmierung. Sie sind in der Regel in der Lage, abhängig von der Problemklasse zu erkennen, welche Programmierparadigmen potenziell für eine softwaretechnische Unterstützung/ Realisierung in Frage kommen. Neben den potenziellen Möglichkeiten sind sie aber auch hinsichtlich der bestehenden Grenzen, Risiken, Notwendigkeiten und Herausforderungen sensibilisiert, was sie dann insgesamt von einer Vielzahl heutiger Manager unterscheidet[10].
Selbstverständlich stellt die für die skizzierten Lehrinhalte zur Verfügung stehende Zeit eine Limitierung dar, insbesondere da im Rahmen der Wirtschaftsinformatik-Lehrveranstaltungen ein breiter, belastbarer Überblick über aktuelle Paradigmen, Technologien etc. vermittelt werden soll. Perspektivisch wäre zu überlegen, ob im Rahmen von BWL-Studiengängen mehr Zeit für Fächer, wie Wirtschaftsinformatik, Digitale Transformation u.ä. zur Verfügung gestellt werden. Könnte dadurch das Zeitvolumen für eigene (ggf. angeleitete) studentische Programmierübungen vergrößert werden, würden sich u.E. weitere positive Effekte, wie z.B. höhere Akzeptanz, besseres Verständnis und Förderung von Kreativität ergeben.

[10] ebenfalls von den BWL-Studierenden, die auch heute noch inhaltlich stark eingeschränkte Wirtschaftsinformatik-Lehrveranstaltungen absolvieren.

5 Fazit und Ausblick

Das Berufsbild des Betriebswirts unterliegt seit einiger Zeit einem drastischen Wandel. Insbesondere durch die allgegenwärtige, „existenzentscheidende“ Digitalisierung und die Notwendigkeit vieler Unternehmen und anderer Gesellschaftsbereiche zur Digitalen Transformation, rücken IT-Themen und –aufgaben unweigerlich in den Arbeitsfokus von Betriebswirten. Einerseits sind sie unter Umständen von der fortschreitenden Digitalisierung selbst betroffen, was zur Notwendigkeit einer berufsinhaltlichen Umorientierung führt, andererseits sind sie potenzielle Entscheidungsträger, Projektmitarbeiter, Projektbegleiter oder Projektmanager in Digitalisierungsprojekten oder in Projekten im Digitalisierungskontext. Dieser Umstand erfordert, dass Betriebswirte ein breites belastbares – in geeigneter Granularität und Tiefe - Wissen auf dem Gebiet der Wirtschaftsinformatik besitzen beziehungsweise erwerben.
Neben dem Wissen um einschlägige aktuelle und zukunftsträchtige IT-Technologien, wie zum Beispiel dem Cloud Computing, IoT, BIG Data usw. sollten Betriebswirte einen belastbaren Überblick über Paradigmen besitzen, die bei der Realisierung IT-gestützter, effizienter und agiler Unternehmensprozesse zum Einsatz kommen können. Zu diesem Wissen gehört auch Wissen über relevante Programmierparadigmen und deren Anwendbarkeit für die Realisierung von Softwarekomponenten, z.B. für die Lösung (oder mindestens Lösungsunterstützung/ Entscheidungsunterstützung) von komplexen betriebswirtschaftlichen Aufgabenstellungen, wobei diese insbesondere im Kontext des Digitalen Unternehmens (vgl. [Jo15], [Jo17]) zusätzlich eine entscheidende Bedeutung besitzen werden. Hierbei bilden Vertreter der deklarativen Programmierung, z.B. die Logische Programmierung und die Constraint (Logische) Programmierung, eine besondere Klasse von Programmierparadigmen. Umso unverständlicher ist es, dass diese bisher nur in wenigen Curricula für die Wirtschaftsinformatik-Lehrveranstaltungen von BWL-Studiengängen Einzug gefunden haben und auch in Standardwirtschaftsinformatik-Lehrbüchern i.d.R. keine nennenswerte Erwähnung finden. Ein Grund könnte sein, dass viele Wirtschaftsinformatikprofessoren und Lehrbuchautoren selbst keine oder nur wenig Erfahrungen und Kenntnisse auf den Gebieten der Logischen Programmierung und Constraint (Logischen) Programmierung besitzen. Das Paper motiviert die Aufnahme entsprechender Lehrinhalte in die BWL-Wirtschaftsinformatik-Lehrveranstaltungen, stellt einige gut geeignete und erprobte Komponenten für die diesbezügliche Lehre vor, nennt Ausbau- und Fortführungsmöglichkeiten und skizziert ein jahrelang praktiziertes Vorgehen, das natürlich dynamisch weiterentwickelt wird/ werden sollte.
Sicherlich ist es in „reinen“ Wirtschaftsinformatik-Studiengängen und wirtschaftsinformatiknahen Studiengängen sehr empfehlenswert, den adressierten Themengebieten einen größeren zeitlichen und inhaltlichen Umfang einzuräumen. Einen besonderen Stellenwert sollten im Kontext der Constraint Programmierung *globale Constraints* und deren Anwendung bei Problemmodellierungen bilden. Ergänzend zur Constraint Logischen Programmierung könnten auch Constraint-Bibliotheken für imperative Programmiersprachen (z.B. CHOCO, JaCoP oder Gecode) und Modellierungssprachen, wie OPL und MiniZinc thematisiert werden. Um den identifizierten Anforderungen gerecht zu werden, gibt es zum Beispiel im Curriculum des HWTK-

Studienganges „Informatik und Management“ gesonderte Kurse, die sich der *Logischen Programmierung*, der *Constraint Programmierung* und der *Künstlichen Intelligenz* widmen.

Literatur

[AW07] Apt, K., Wallace, M.: Constraint Logic Programming using ECLIPSE, Cambridge University Press, 2007.

[Do15] Domschke, W. et al: Einführung in Operations Research. 9. Auflage, Springer Gabler, 2015.

[He09] Heinrich, W.: Geschichte und Anwendung von Magischen Quadraten in Zeit und Raum, Bohmeier, 2009.

[Jo15] John, U.: Digitales Unternehmen – Bausteine für Effizienz, Agilität und Transparenz. In (Cunningham, D. et al (Ed.)) Proc. INFORMATIK 2015, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, 2015.

[Jo17] John, U.: Deklarative Unternehmensmodelle – essentielle Bausteine Digitaler Unternehmen für optimierende Reorganisationen. INFORMATIK 2017 (Maximilian Eibl, Martin Gaedke (Ed.)), Lecture Notes in Informatics (LNI), Gesellschaft für Informatik 2017.

[La16] Laudon, K. C., Laudon, J. P., Schoder, D.: Wirtschaftsinformatik. 3. Auflage, Pearson, 2016.

[La18] Laudon, K. C., Laudon, J. P.: Management Information Systems - Managing the Digital Firm. 15. Auflage, Pearson, 2018.

[Me17] Mertens, P. et al: Grundzüge der Wirtschaftsinformatik. 12. Auflage, Springer Gabler, 2017.

[Si05] Simonis, H.: Sudoku as Constraint Problem. Working Notes of the CP 2005 Workshop on Modeling and Reformulating Constraint Satisfaction Problems.