V "Wissenschaftliches Arbeiten i. d. Informatik"

Gutes wissenschaftliches Schreiben

Prof. Dr. Lutz Prechelt
Freie Universität Berlin, Inst. f. Informatik

- Qualitätskriterien, Ziele
- 1. Äußere Form
- 2. Ausdrucksweise
  - präzise, unmehrdeutig, neutral, klar, logisch korrekt
- 3. Inhalt
  - Abstract, Einleitung, Forschungsfragen, Begriffe, Beiträge, verwandte Arbeiten, Methodik, Faktenangaben, Schlussfolgerungen
- Technik: LaTeX, BibTeX

Qualitätsmerkmale guter wiss. Texte

1. Äußere Form:
   - Layout nach Vorgaben ●●●
   - Zitationsfähig ●●●
2. Ausdrucksweise:
   - Präzise, nicht mehrdeutig ●●
   - Sachlich, neutral ●●●
   - Einfach und klar ●●
   - Logisch korrekt, nicht umgangssprachlich ●●●
   - Titel ist informativ ●●
3. Inhalt:
   - Abstract fasst Inhalt zusammen ●
   - Erläutert Hintergrund ●●●
   - Definiert Kernbegriffe ●●
   - Benennt Forschungsfragen ●●●
   - Benennt eigene Beiträge ●●
   - Charakterisiert verwandte Forschungsarbeiten ●
   - Erklärt Methodik ●●
   - Fakten sind korrekt ●●●
   - Behauptungen sind belegt ●●
   - Relevante Fakten genannt ●●
   - Schlussfolgerungen nach-vollziehbar und vorsichtig ●●

1. Äußere Form: Layout nach Vorgaben ●●●

- Der Erscheinungsort gibt ein Layout präzise vor, das genau einzuhalten ist
  - bei Zeitschriften sorgt die Zeitschrift dafür meist am Ende selbst
  - bei Konferenzen sind die Autoren allein zuständig

884 IEEE TRANSACTIONS ON COMPUTERS, VOL. 64, NO. 4, APRIL 2015

A Cache Hierarchy Aware Thread Mapping Methodology for GPGPUs

Bo-Cheng Charles Lai, *Member, IEEE*, Hsien-Kai Kuo, *Student Member, IEEE*, and Jing-Yang Jou, *Fellow, IEEE*

Abstract—The recently proposed GPGPU architecture has added a multi-level hierarchy of shared cache to better exploit the data locality of general purpose applications. The GPGPU design philosophy allocates most of the chip area to processing cores, and thus results in a relatively small cache shared by a large number of cores when compared with conventional multi-core CPUs. Applying a proper thread mapping scheme is crucial for gaining from constructive cache sharing and avoiding resource contention among thousands of threads. However, due to the significant differences on architectures and programming models, the existing thread mapping approaches for multi-core CPUs do not perform as effective on GPGPUs. This paper proposes a formal model to capture both the characteristics of threads as well as the cache sharing behavior of multi-level shared cache. With appropriate proofs, the model forms a solid theoretical foundation beneath the proposed cache hierarchy aware thread mapping methodology for multi-level shared cache GPGPUs. The experiments reveal that the three-staged thread mapping methodology can successfully improve the data reuse on each cache level of GPGPUs and achieve an average of 2.3× to 4.3× runtime enhancement when compared with existing approaches.

Index Terms—Multithreaded processors, cache memories, shared memory, performance analysis and design aids

1 Introduction

While Moore's law continues to double the transistor density at every technology generation, the emerging processor architecture has shifted to the chip multiprocessors (CMPs) design. CMPs exploit the thread-level-parallelism (TLP) with multiple computing cores on a chip, and have the potential to achieve superior performance and energy efficiency. General-purpose graphic processing units (GPGPUs) are one of the widely used and aggressive designs of CMPs [1], [2]. Instead of integrating conventional CPUs and huge caches on a chip, GPGPUs allocate most of the chip area to hundreds of simpler and smaller cores [3]. By concurrently executing thousands of threads, GPGPUs have the ability to enable massive TLP of a wide range of applications [4], [5].

Even with the ability to execute thousands of threads, the peak performance of GPGPUs is usually confined by insufficient data access bandwidth for parallel processing [6]. Adding *multi-level shared cache* and enabling *memory coalescing* are the two techniques adopted by recent GPGPUs to enhance the data access bandwidth [1], [2]. The on-chip cache in a GPGPU proactively keeps the reusable data within the chip to facilitate fast and efficient accesses. Moreover, the on-chip processing cores can share the cache at different memory hierarchy levels to exploit the *data reuse* among concurrent threads. In GPGPUs, the shared cache is further implemented with the coalescing technique [7], which is able to respond to a group of sharing cores concurrently and thus reduces the number of transactions to the shared cache.

However, a parallel program cannot always take full advantages of these techniques. The actual performance benefits are significantly decided by the data sharing behavior among threads and how the threads are mapped to the processing cores in a GPGPU. A parallel program running on a GPGPU is said to have the *constructive cache sharing* effect [8] when the behavior of threads can exploit the above mentioned beneficial effects. Unfortunately, an arbitrary mapping of threads could easily suppress the potential benefits. Even worse, the un-coordinated threads could generate a series of *destructive cache sharing* effects [8] that impair the overall performance. For example, a careless mapping could cause serious *cache contention* by forcing a large number of threads to compete for a small cache and flush the useful data of each other. This could significantly degrade the performance of massive TLP on GPGPUs.

Maximizing the benefit of constructive cache sharing on a GPGPU with multi-level shared cache is not a trivial task. Both the constructive and destructive cache sharing effects could dispersedly exist in different levels [9]. The problem becomes even more complex when dealing with an application that generates irregular data access patterns. Unlike regular applications, the data sharing in an irregular application behaves non-uniformly across different threads [9], [10], [11], [12]. The irregularity causes non-intuitive execution behavior, and therefore raises design challenges.

Most of the existing works [13], [14], [15], [16] address the optimization issues of the multi-level shared cache by mapping threads to particular cores for a given multi-core

- B.-C.C. Lai and H.-K. Kuo are with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, ROC. E-mail: bcla@mail.nctu.edu.tw, hkkuo@ieda.ee.nctu.edu.tw.
- J.-Y. Jou is with the Department of Electrical Engineering, National Central University, Taoyuan 320, and the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, ROC. E-mail: jyjou@ee.ncu.edu.tw, jyjou@faculty.nctu.edu.tw.

Manuscript received 4 Apr. 2013; revised 22 Nov. 2013; accepted 29 Dec. 2013. Date of publication 24 Feb. 2014; date of current version 13 Mar. 2015. Recommended for acceptance by D.A. Bader.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2308179

0018-9340 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

IEEE Transactions on Computers (erscheint seit 1968)

Approximate Verification of the Symbolic Dynamics of Markov Chains

2

MANINDRA AGRAWAL, Indian Institute of Technology Kanpur, India
S. AKSHAY, Indian Institute of Technology Bombay, India
BLAISE GENEST, CNRS, UMR IRISA, Rennes, France
P. S. THIAGARAJAN, National University of Singapore

A finite-state Markov chain M can be regarded as a linear transform operating on the set of probability distributions over its node set. The iterative applications of M to an initial probability distribution μ_0 will generate a trajectory of probability distributions. Thus, a set of initial distributions will induce a set of trajectories. It is an interesting and useful task to analyze the dynamics of M as defined by this set of trajectories. The novel idea here is to carry out this task in a *symbolic* framework. Specifically, we discretize the probability value space $[0, 1]$ into a finite set of intervals $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$. A concrete probability distribution μ over the node set $\{1, 2, \ldots, n\}$ of M is then symbolically represented as D, a tuple of intervals drawn from $\mathcal{I}$ where the ith component of D will be the interval in which $\mu(i)$ falls. The set of discretized distributions $\mathcal{D}$ is a finite alphabet. Hence, the trajectory, generated by repeated applications of M to an initial distribution, will induce an infinite string over this alphabet. Given a set of initial distributions, the symbolic dynamics of M will then consist of a language of infinite strings L over the alphabet $\mathcal{D}$.

Our main goal is to verify whether L meets a specification given as a linear-time temporal logic formula φ. In our logic, an atomic proposition will assert that the current probability of a node falls in the interval I from $\mathcal{I}$. If L is an ω-regular language, one can hope to solve our model-checking problem (whether $L \models \varphi$?) using standard techniques. However, we show that, in general, this is not the case. Consequently, we develop the notion of an ϵ-approximation, based on the transient and long-term behaviors of the Markov chain M. Briefly, the symbolic trajectory ξ' is an ϵ-approximation of the symbolic trajectory ξ iff (1) ξ' agrees with ξ during its transient phase; and (2) both ξ and ξ' are within an ϵ-neighborhood at all times after the transient phase. Our main results are that one can effectively check whether (i) for each infinite word in L, at least one of its ϵ-approximations satisfies the given specification; (ii) for each infinite word in L, all its ϵ-approximations satisfy the specification. These verification results are strong in that they apply to all finite state Markov chains.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods, model checking*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Probabilistic computation*

General Terms: Theory, Verification, Algorithms

Additional Key Words and Phrases: Markov chains, discretization, LTL logic, approximate model checking

Part of this work was done while M. Agrawal was visiting the IMS Workshop on Automata Theory at National University of Singapore. Support from the workshop and J. C. Bose Fellowship is gratefully acknowledged. S. Akshay would also like to acknowledge partial support from DST-INSPIRE faculty award [IFA12-MA-17] and ANR project ImpRo (ANR-2010-BLAN-0317). B. Genest acknowledges partial support from ANR projet STOCH-MC (ANR-13-BS02-0011-01) and EU project DISC. P. S. Thiagarajan acknowledges partial support from the Singapore Ministry of Education grant MOE2011-T2-2-012.
Authors' addresses: M. Agrawal, Department of CSE, IIT Kanpur, India; email: manindra@iitk.ac.in; S. Akshay, Department of CSE, IIT Bombay, Mumbai, India; email: akshayss@cse.iith.ac.in; B. Genest, CNRS, UMR IRISA, Campus de Beaulieu, Rennes, France; email: bgenest@irisa.fr; P. S. Thiagarajan, School of Computing, National University of Singapore, Singapore; email: thiagu@comp.nus.edu.sg.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2015 ACM 0004-5411/2015/02-ART2 $15.00
DOI: http://dx.doi.org/10.1145/2629417

Journal of the ACM (erscheint seit 1954)

Freie Universität Berlin

Bestimmte Angaben auf Titelblatt nötig:

- Man muss wiss. Artikel eindeutig identifizieren können
  - → Titel
  - → genauer Erscheinungsort
    - Konferenz: z.B. Name, Kürzel, Jahr, Organisation/Verlag
      - 7th IFIP Wireless and Mobile Networking Conf. (WMNC), 2014
    - Zeitschrift: z.B. Name, Jahr, Monat, Band, Nummer, Seitenzahlen
      - Journal of the ACM, Vol. 62, No. 1, Article 2, February 2015
      - IEEE Transactions on Computers, Vol. 64, No. 4, April 2015
- Die Veröffentlichungen sind f. eine/n Wissenschaftler/in die wichtigsten Karrierebausteine
  - → Autoren
    - → ORCID-ID (z.B. jede/r fünfte Koreaner/in heißt "Kim")

2. Ausdrucksweise: Präzise, nicht mehrdeutig ●●

- Hohe Aufmerksamkeit auf Vermeidung von Mehrdeutigkeit
  - Leser aus versch. Kulturen
  - begrenzte Englischkenntnisse
  - Zeitlicher Wandel von Selbstverständlichem

- PrePep14, Sect. 6.1:
  - "draw a stratified random sample of 100 file changes as follows:
    (a) 25 file changes from files with at most 3 lines changed;
    (b) 25 file changes from files with more than 3 lines changed;
    (c) 25 file changes from one file from a commit with at most 8 lines changed;
    (d) 25 file changes from one file from a commit with more than 8 lines changed."

2. Ausdrucksweise: Sachlich, neutral ●●●

- Wertungen vermeiden, wo möglich
  - z.B. statt "accuracy is bad" lieber "insufficiently accurate"
    - wenn Maßstab vorhanden
  - lieber "incorrect" als "wrong", denn letzteres hat einen moralischen Unterton
  - usw.
- Wichtungen vermeiden, wenn es keine klaren Maßstäbe dafür gibt
  - z.B. kommt das Wort "very" nur selten vor.

- PrePep14, Sect. 9:
  - "The original authors of the SZZ algorithm, Sliwerski et al. [25], were well aware that not all of the changes the algorithm points out are defect insertions; they therefore called the outputs ‘‘fix-inducing changes’’ instead. Unfortunately, this term is still misleading: If the output of the algorithm is incorrect (not a rare case as we have seen), the particular changes found did not ‘‘induce’’ the bugfix."

2. Ausdrucksweise: Einfach und klar ●●

- Gedankengang möglichst geradlinig anlegen
  - bei komplizierter Sachlage ganz schön schwierig!
- Komplizierte Sätze in mehrere einfache zerlegen
- Wichtige Unterscheidungen oder Alternativen gelegent-lich in Erinnerung rufen
- Komplizierte Sachverhalte zusätzlich nochmal vereinfacht aufschreiben

- PrePep14, Sect. 2.1:
  - "A defect correction is a bugfix whose bugtracker entry describes a defect (rather than an issue)."
- PrePep14, Sect. 6:
  - "Once the defect-related bugfix links have been isolated, the next step is the DCDIM: mapping the defect correction (‘‘bug fix’’) to the corresponding defect insertion (‘‘bug commit’’)."
    - und dann geht es so weiter (komplizierter Satz schön zerlegt):

2. Ausdrucksweise: Einfach und klar •• (2)

"The basic approach of DCDIM is simple and consists of four steps:

- S1: consider which lines have been changed in a bugfix,
- S2: assume all of these lines (but only these lines) to be defective,
- S3: trace backwards through the version history to identify for each of these lines the last commit that has changed the line, and
- S4: treat each such commit to be a defect insertion.

Unfortunately, every single one of these steps has problems and can go wrong:

- P1: the bugfix change may have touched other lines than only the lines of the actual defect correction,
- P2: the defect correction itself may also have changed more lines than just the defective ones, e.g. in order to provide a cleaner fix,
- P3: a line change in the history may leave the program behavior unchanged,
- P4: the actual defect insertion may have been earlier and the last change is in fact unrelated to the defect."

2. Ausdrucksweise: Einfach und klar •• (3)

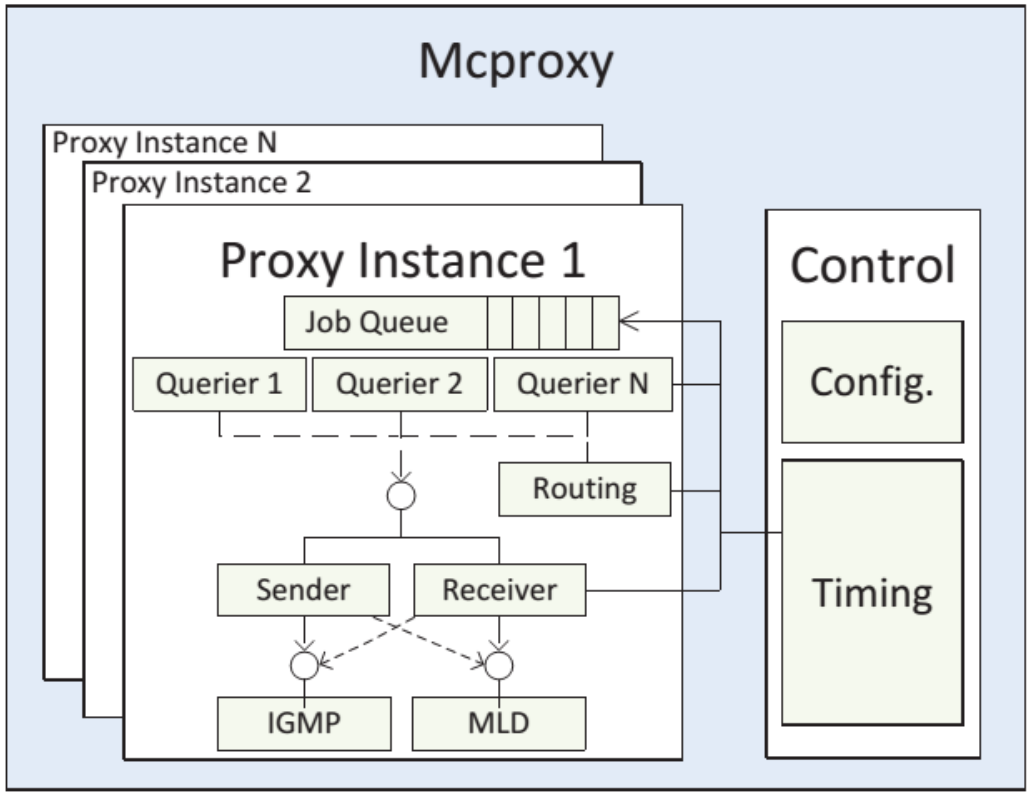
Freie Universität Berlin

- Tabellen, Tabellen!

Table I. LINK DELAYS AND VARIATIONS IN THE TESTBED

Links	Delay (ms)	± Jitter (ms)
UMTS ((M)N ⟷ MAG)	200	± 50
LTE ((M)N ⟷ MAG)	5	± 3
Tunnel (MAG ⟷ LMA)	20–40	± 6
Peering (MAG ⟷ MAG)		
Backbone Path (LMA ⟷ LMA)		

- Abbildungen, Abbildungen!



Beispiele aus "Peer my Proxy"

- Sowas bitte nicht:
  - "Hertha hat gewonnen,
    weil die Fans hinterher in der Stadt so gesungen haben."
- sondern mindestens so:
  - "Ich glaube, Hertha hat gewonnen,
    weil die Fans hinterher in der Stadt so gesungen haben."
- aber eigentlich präziser bitte so:
  - "Die Fans haben hinterher in der Stadt so gesungen;
    deshalb glaube ich, Hertha hat gewonnen."

- Der Titel sollte das Thema möglichst genau umreißen
  - noch besser: sogar das Resultat
  - ohne dabei gar zu lang zu werden (üblich: ca. 6 bis 12 Wörter)

- In der theoretischen Informatik klappt das oft sehr gut:
  - Solving the Weighted Stable Set Problem in Claw-Free Graphs via Decomposition
  - Satisfiability Allows No Nontrivial Sparsification unless the Polynomial-Time Hierarchy Collapses

- Anderswo geht das meist nur schwierig:
  Mangel an passender Terminologie

3. Inhalt: Die übliche Gesamtstruktur

- "Abstract":
  - Zusammenfassung des Inhalts
    - wird am häufigsten gelesen
    - max. Länge oft vorgegeben
      - z.B. 150 Wörter
- "Introduction":
  - Themengebiet, Motivation, Begriffe
  - Forschungsfrage, Relevanz, Beiträge
  - verwandte Arbeiten
    - evtl. separater Abschnitt
    - evtl. erst nach Hauptteil
  - Überblick über Struktur
- Hauptteil (variabel):
  - Methoden, Techniken, Entwurf, Vorgehen u.ä.
  - Resultate
  - Diskussion
- "Conclusion":
  - Schlussfolgerungen aus Resultaten
    - kaut oft nur Abstract wieder
  - Nächste Forschungsschritte
- "References"
  - Liste der Quellen
- Appendices:
    - (eher selten)
  - Detailinformationen

3. Inhalt:
Abstract fasst Inhalt zusammen

- "Abstract" heißt auf Deutsch "Zusammenfassung"
  - Soll komprimiert möglichst den ganzen Artikel wiedergeben:
    - Thema, Methoden, Resultate
- Sehr verbreitete Unsitte: Der Abstract kündigt nur an.
  - Negativbeispiel 1
    - schauderlich
  - Negativbeispiel 2
    - schon merklich besser
  - Positivbeispiel 1
  - Positivbeispiel 2
- Üblich in Naturwissenschaften: Strukturierte Abstracts
  - Mit festen Abschnitten: Context/Background, Question/Objective, Method, Results, Conclusion
  - Das ist tendenziell besser
- Für empirische Artikel üblicher werdend
  - mancherorts sogar verlangt (hier: *Information and Software Technology*)

- Alle Nichtexperten brauchen eine Themeneinführung:
  - Wo spielt der Artikel?
  - Woher kommt das Problem?
  - Warum ist es von Interesse?
  - Was sind übliche Verfahren? Gut gesichertes Wissen?

- PrePep14, 1. Absatz:

  "Mining software repositories (MSR) is a set of techniques that exploit the data stored in [...] source code version archives or issue tracking databases in order to obtain relevant insights [...].

One of the potentially useful application areas [...] is understanding defect insertion [...] processes. [...] defect insertion circumstance analysis (DICA) attempts to identify correlates of defect insertions: when, where (contexts), and how they happen and who makes them happen. Such insights can potentially be used to make valuable process improvements."

- Präzise Aussagen benötigen Terminologie mit klarer Bedeutung.
- Jeder Artikel muss seine Hauptbegriffe selbst definieren
  - damit die Bedeutung zweifelsfrei ist.

- PrePep14, Sect. 2.1:
  - "A *defect* is a property of source code that triggers avoidable rework (possibly outside the observed timeframe).
  - *Rework* is any modification performed on a section of program source code that was written and checked in earlier. […] We call rework *avoidable* if the need to make that change was in principle known at the time of the original work. We call rework *unavoidable* if that need arose only later […]."

- Explizite Nennung der Frage erleichtert das Verständnis sehr.
  - Bei konstruktiven (anstatt empirischen) Artikeln leider oft nur implizit angegeben.

- PrePep14, Sect. 1.1:
  - "The present article is a case study aimed at the research question formulated in the title: Why aren't MSR techniques used for DICA more often than they are?"

3. Inhalt:
Benennt eigene Forschungsbeiträge ••

- Explizite Benennung, was die Autor/inn/en für neu halten, erleichtert das Verständnis sehr
  - Was trägt dieser Artikel gegenüber dem bisherigen Stand der Wissenschaft bei?

- PrePep14, Sect. 1.2:

  "[…] Our article makes the following […] contributions:
  - It presents results of a 4-person-month DICA attempt for a large […] industrial software repository performed by practioners […]. We are not aware of any other such report from an explicitly described industrial setting.
  - It sorts out, at least roughly, the relative contributions of five different potential reasons (R2–R6) for not using DICA more often.
  - It demonstrates the dominant weight of the two reasons mentioned above."

- Zum Gesamtverständnis ist wichtig:
  - Was für ähnliche oder verwandte Arbeiten ("related work") gibt es schon?
  - Was ist dort gleich/anders beim Vorgehen?
    - Hieraus ergibt sich auch die Originalität
  - Was ist dort gleich/anders bei den Ergebnissen?
- Meist wird leider nur ein Unterscheidungsmerkmal angegeben
  - keine ernsthafte Vergleichsdiskussion

- PrePep14, Sect. 9:
  - "[Für DICA muss man A, B, C lösen.] Reading the DCDIM/DICA literature, it quickly becomes clear that A receives not much attention and C receives almost none at all.
    We will therefore review the literature and report weaker aspects [...]. We mark up the discussion with letters A, B, C to indicate which of the difficulties it pertains to."

3. Inhalt:
Erklärt Methodik ●●

- Ein Artikel ist wertlos, wenn man nicht weiß, wie seine Ergebnisse zustande kommen.
- Leider fehlt hier oft Detail.
  - Technical Reports sind aus der Mode gekommen
  - Anhänge noch nicht genügend in Mode
  - Manchmal steht was in Dissertationen.

PrePep14, Überschriften:

- 3. Design and method of the case study
  - 3.1. The case: DICA for Infopark CMS Fiona
  - 3.2. Infopark's DICA quality criteria
  - 3.3. Propositions: possible reasons why DICA is rare
  - 3.4. Sources of evidence and use of triangulation
  - 3.5. Why just a single case?
  - 3.6. Structure of the case presentation

- Wichtiges Merkmal wiss. Artikel:
  - Alle Faktenangaben sind zutreffend und verlässlich
  - Alle unsicheren Angaben sind als solche gekennzeichnet
    - insbes. Interpretationen, Schätzungen, Vermutungen
- PrePep14, Sect. 5:
  - "[…] there is a large fraction (perhaps a majority) of change requests that lies in between [clear defects and clear non-defects]"
- PrePep14, Sect. 10.1:
  - "Note that the high quality of the Infopark data with respect to bugfix links [18] suggests that the non-minimal corrections are not simply a sign of low development discipline. Rather, we conjecture the effect emerges from application domain and business context"

3. Inhalt: Behauptungen sind belegt ●●

- Üblich ist:
  - Leser glauben die Fakten
    - wenn die Methodik sie hergibt
  - Alles andere muss belegt werden
- Zwei Möglichkeiten:
  - *Eigene* Daten oder Beweise
    - Beleg durch Aufschlüsselung, Argumentation, Rechnung, Theorem etc.
  - Verweis auf *andere* wiss. Veröffentlichungen
    - Beleg durch Zitation (Quellenangabe)

- PrePep14, Sect. 6:
  - "P2: the defect correction itself may also have changed more lines than just the defective ones, e.g. in order to provide a cleaner fix[.]
  - With respect to P1 and P2, Herzig and Zeller [15] report for five open source projects that between 7% and 20% of the defect correction changes address multiple concerns at once (‘‘tangled changes’’) rather than just the defect correction concern."

3. Inhalt:
Relevante Fakten werden genannt ●●

- Alle Informationen, die ein/e Leser/in für wichtig halten könnte, sollten genannt sein
  - Also viele
  - Nicht klar, welche
    - Also noch ein paar mehr…

- PrePep14, Sect. 4.1:

  "[We] started at a version archive […] of 11 years of development (more than 45,000 commits) and a Bugzilla database from 8 years of development (9444 entries […]) in which 5005 bugfix link candidates had been identified, which refered to 4499 different commits and 3203 different Bugzilla entries. These candidates were found by simply collecting all potential ID reference numbers found anywhere in commit [msgs], bug entry descriptions, and bug entry comments […]."

- Hier oder im Hauptteil:
  - Diskussion des Grades an Gültigkeit
- Unsitte: "Conclusions" wiederholen ungefähr den Abstract
- Richtig wäre: Kenntnis aller Details voraus-setzen und über die puren Resultate hinaus schlussfolgern

- PrePep14, Sect. 8.1:
  - "The problem from mixing defects and issues (difficulty A, step 2) will be smaller if the organization applies a suitable definition of defect more consistently. It could also be larger, though, if [...]"
- PrePep14, Sect. 10.1:
  - "Overall, the low reliability (R6) achieved for the DICA results, reinforced by the lack of a practical assessment method for that reliability (R5), are clearly the dominant reasons why Infopark dropped its DICA initiative and we see little reason why many other organizations should come to a different conclusion."

Das waren die Qualitätsmerkmale guter wiss. Texte

Freie Universität Berlin

1. Äußere Form:
   - Layout nach Vorgaben ●●●
   - Zitationsfähig ●●●
2. Ausdrucksweise:
   - Präzise, nicht mehrdeutig ●●
   - Sachlich, neutral ●●●
   - Einfach und klar ●●
   - Logisch korrekt, nicht umgangssprachlich ●●●
   - Titel ist informativ ●●
3. Inhalt:
   - Abstract fasst Inhalt zusammen ●
   - Erläutert Hintergrund ●●●
   - Definiert Kernbegriffe ●●
   - Benennt Forschungsfragen ●●●
   - Benennt eigene Beiträge ●●
   - Charakterisiert verwandte Forschungsarbeiten ●
   - Erklärt Methodik ●●
   - Fakten sind korrekt ●●●
   - Behauptungen sind belegt ●●
   - Relevante Fakten genannt ●●
   - Schlussfolgerungen nachvollziehbar und vorsichtig ●●

"Aha, und dann erscheint mein Artikel?"

- Die obigen Kriterien sind alles eher Formmerkmale
- Inhaltlich kommt es noch auf "originality" an und auf "relevance".
- Wo steht das?:
  - Konferenz: Call for Papers
    - z.B. ICSE 2015
  - Zeitschrift: Scope, Mission, Author Guide
    - z.B. IEEE Transactions on Dependable and Secure Computing
      - die wollen z.B. keine rein theoretischen Kryptographie-Artikel

- Außerdem leider: Starker Drang, einen guten (Nützlichkeit!) Eindruck machen zu wollen
  - Negative Resultate werden weniger gern aufgenommen
  - Siehe die Begutachtungs-karriere von PrePep14

- Man muss die Gutachter/innen überzeugen
  - Dazu gibt es diverse Tipps
    - (z.T. themenabhängig)
  - z.B. diese:

Kent Beck: How to Get a Paper Accepted at OOPSLA

- OOPSLA (heute SPLASH) ist eine Top-Konferenz im Bereich Sprachen, Entwurf und Programmierung
- Du hast 30 Sekunden, um die Aufmerksamkeit zu fesseln
- Zuständig dafür ist der Abstract
- Geeignete Struktur (4 Sätze):
  1. Was ist das Problem?
  2. Warum ist es ein Problem?
  3. **Überraschende Behauptung**
  4. Auswirkungen

- Der Rest muss die überraschende Behauptung belegen/verteidigen
- Geeignete Struktur (4 Abschnitte):
  1. Das Problem (Was? Warum? Warum wichtig?)
  2. Die Lösung
  3. Warum die Lösung klappt
  4. Was andere da bisher so hinbekommen haben

- Geeignet für Systemartikel
  - weniger für empirische und theoretische

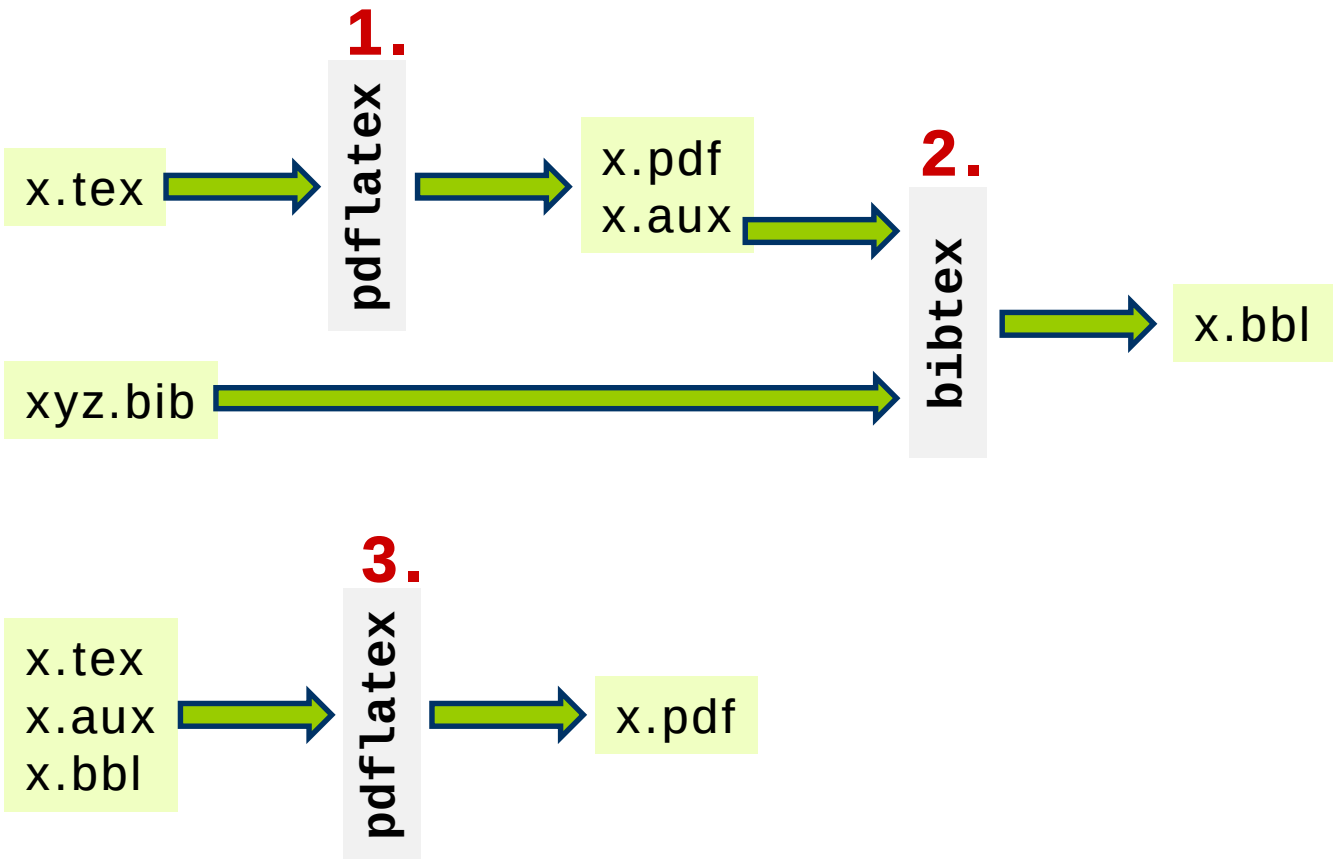
Technik: LaTeX, BibTeX

- Informatik-Artikel werden in der Regel mit LaTeX und BibTeX geschrieben
  - nicht mit MS Word
- Benutzung mit
  - Texteditor + Kommandozeile

  oder
  - TeX-IDE

  Liste von LaTeX-Werkzeugen
- Funktionsprinzip LaTeX:
  - Quelltext mit Markup
  - Textprozessor erzeugt daraus PDF
- Funktionsprinzip BibTeX:
  - Literaturdatenbank als Textdatei
  - BibTeX-Prozessor entdeckt Referenzierungstellen und erzeugt Literaturliste
    - Stil ist wählbar

LaTeX, BibTeX: grober Datenfluss



LaTeX, BibTeX: Vorteile/Nachteile

- Stärken:
  - Funktioniert stabil
  - Fokus auf Inhalt statt auf Form
  - Auf jedem Betriebssystem
  - Gutes, einheitliches Layout
    - anstatt Handgefummel.
    - Einfaches Ändern des Layouts bei Neueinreichung
    - insbes. auch für Literaturliste
  - Literaturdatenbank simpel, robust, langlebig
  - Hohe Flexibilität
    - viele Erweiterungspakete
    - Makros
    - Manipulation mit Skripten
- Schwächen:
  - Etwas Lernaufwand
  - Für kleine Sachen umständlich
  - Quelltext weniger übersichtlich
  - Viele Dateien, Werkzeuge und Erweiterungspakete

LaTeX, BibTeX: Beispiel

- Wir schauen uns ein Beispiel an:
  - Quelldatei beispiel.tex
    - plus Literaturdatenbank vwiss.bib
  - Ergebnis beispiel.pdf

- Werkzeuge:
  - Texteditor GNU Emacs
  - PDF-Viewer SumatraPDF
    - der aktualisiert automatisch
  - Cygwin bash
    - Shell für Kommandozeile

Weitere Quellen

- Tim Skern: Writing Scientific English
  - Lehrbuch zur Sprachebene
- Jonathan Shewchuk: Three Sins of Authors in Computer Science and Math, 1997.
  - Berechtigtes Geschimpfe über drei verbreitete Marotten; ziemlich saftig formuliert.

Danke!