Jana Dunfield

School of Computing
Queen's University
Goodwin Hall Room 557
Kingston, ON K7L 3N6, Canada

jana@cs.queensu.ca
604–367–4987
613–513–3166
http://research.cs.queensu.ca/∼jana

Some publications and credentials used the name Joshua Dunfield.

Research interests

Programming languages: type inference and type-based software verification; incremental computation; functional programming; automated deduction.

Teaching experience

Functional programming (Carnegie Mellon; McGill; Queen's); formal logic and logic programming (Queen's); programming languages (Carnegie Mellon; McGill; UBC; Queen's); software quality assurance (Queen's).

Personal

US and Canadian citizen.

Positions

Assistant Professor, School of Computing, Queen's University (Kingston, Ontario), Aug. 2017–present.

Sessional Lecturer, Department of Computer Science, University of British Columbia (Vancouver), September–Dec. 2016, Sept.–Dec. 2015.

Research Associate (non-tenure-track faculty), Department of Computer Science, University of British Columbia (Vancouver), May 2014–May 2017.

Postdoctoral Researcher, Max Planck Institute for Software Systems (Kaiserslautern and Saarbrücken, Germany), Sept. 2010–May 2014.

Postdoctoral Fellow, School of Computer Science, McGill University, Sept. 2007–Aug. 2010.

Course Lecturer, School of Computer Science, McGill University, Jan.–April 2010.

Teaching Assistant, School of Computer Science, McGill University, Jan.–March 2008.

Education

Ph.D. Computer Science, Carnegie Mellon University (Pittsburgh, PA, USA), August 2007.
Dissertation title: *A Unified System of Type Refinements.*
Committee: Frank Pfenning (chair), Jonathan Aldrich, Robert Harper, Benjamin Pierce (University of Pennsylvania).
US National Science Foundation Graduate Research Fellowship, 2000–2003.

B.S. (high honors) Computer Science, Portland State University (Portland, Oregon, USA), August 2000.

Grants and awards

Queen's University Research Initiation Grant, C$60,000 (total; awarded 2017).

Programming languages for scalable incremental computation and advanced gradual typing. NSERC Discovery Grant, C$177,500 (total; 2018–2023).

PPDP 2008 "Most Influential Paper" (10-year test of time) award (with Brigitte Pientka) for "Programming with proofs and explicit contexts", received September 2018.

Journal and conference papers

Jana Dunfield and Neelakantan R. Krishnaswami. Sound and complete bidirectional typechecking for higher-rank polymorphism with existentials and indexed types. arXiv:1601.05106 [cs.PL]. *46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL '19)*. (77/267 ≈ 28% accepted.)

Jana Dunfield. Extensible datasort refinements. In *European Symposium on Programming (ESOP '17)*, pp. 476–503, April 2017. (36/112 ≈ 32% accepted.)

Khurram A. Jafery and Jana Dunfield. Sums of uncertainty: refinements go gradual. In *44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL '17)*, pp. 804–817, 2017. (64/282 ≈ 23% accepted.)

continued on next page

Journal and conference papers (cont.)

Matthew A. Hammer, Jana Dunfield, Kyle Headley, Nicholas Labich, Jeffrey S. Foster, Michael Hicks and David Van Horn. Incremental computation with names.
In *ACM SIGPLAN Int'l Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2015. (53/210 ≈ 25% accepted.)
Implementation accepted by the OOPSLA AEC (Artifact Evaluation Committee).

Jana Dunfield. Elaborating evaluation-order polymorphism.
In *20th ACM SIGPLAN Int'l Conference on Functional Programming (ICFP '15)*, 2015. (35/119 ≈ 29% accepted.)

Jana Dunfield.
Elaborating intersection and union types.
Journal of Functional Programming 24(2–3), pp. 133–165, 2014 (Cambridge Univ. Press).

Yan Chen, Jana Dunfield, Matthew A. Hammer and Umut A. Acar.
Implicit self-adjusting computation for purely functional programs.
Journal of Functional Programming 24(1), pp. 56–112, 2014 (Cambridge Univ. Press).

Jana Dunfield and Neelakantan R. Krishnaswami.
Complete and easy bidirectional typechecking for higher-rank polymorphism.
In *18th ACM SIGPLAN Int'l Conference on Functional Programming (ICFP '13)*, pp. 429–441, Sept. 2013. (40/133 ≈ 30% accepted.)

Jana Dunfield.
Elaborating intersection and union types.
In *17th ACM SIGPLAN Int'l Conference on Functional Programming (ICFP '12)*, pp. 17–28, Sept. 2012. (32/88 ≈ 36% accepted.)
Selected for an invitation to submit an extended version to *J. Functional Programming*.

Yan Chen, Jana Dunfield and Umut A. Acar.
Type-directed automatic incrementalization.
In *33rd ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI '12)*, pp. 299–310, June 2012. (48/255 ≈ 19% accepted.)

Yan Chen, Jana Dunfield, Matthew A. Hammer and Umut A. Acar.
Implicit self-adjusting computation for purely functional programs.
In *16th ACM SIGPLAN Int'l Conference on Functional Programming (ICFP '11)*, pp. 129–141, September 2011. (33/92 ≈ 36% accepted.)

Brigitte Pientka and Jana Dunfield.
Beluga: a framework for programming and reasoning with deductive systems (System Description).
In *Int'l Joint Conference on Automated Reasoning (IJCAR '10)*, pp. 15–21, 2010. (40/89 ≈ 45% accepted.)

Brigitte Pientka and Jana Dunfield.
Programming with proofs and explicit contexts.
In *10th Int'l ACM Symposium on Principles and Practice of Declarative Programming (PPDP '08)*, pp. 163–173, 2008. (24/51 ≈ 47% accepted.) "Most Influential Paper" award, received at PPDP 2018.

Jana Dunfield and Frank Pfenning. Tridirectional typechecking.
In *31st ACM Symposium on Principles of Programming Languages (POPL '04)*, pp. 281–292, 2004. (29/176 ≈ 16% accepted.)

Jana Dunfield and Frank Pfenning.
Type assignment for intersections and unions in call-by-value languages.
In *6th Int'l Conf. on Foundations of Software Science and Computation Structures (FoSSaCS '03)*, pp. 250–266, 2003. (26/94 ≈ 28% accepted.)

Workshop papers (refereed)

Jana Dunfield. Annotations for intersection typechecking.
In pre-proceedings of *Workshop on Intersection Types and Related Systems (ITRS '12)*, June 2012; post-proceedings in *EPTCS* 121, 2013, pp. 35–47.

Jana Dunfield. Untangling typechecking of intersections and unions.
In: pre-proceedings of *Workshop on Intersection Types and Related Systems (ITRS '10)*, pp. 59–70, 2010; post-proceedings in *EPTCS* 45, 2011.

Jana Dunfield. Greedy bidirectional polymorphism.
In *ACM SIGPLAN Workshop on ML (ML '09)*, pp. 15–26, 2009.

Jana Dunfield and Brigitte Pientka. Case analysis of higher-order data.
In *Int'l Workshop on Logical Frameworks and Meta-languages: Theory and Practice (LFMTP '08)*, pp. 69–84, 2008. Elsevier *ENTCS* 228.

Jana Dunfield. Refined typechecking with Stardust.
In *Workshop on Programming Languages meets Program Verification (PLPV '07)*, pp. 21–32, 2007.

Dissertation

Jana Dunfield. *A Unified System of Type Refinements*.
Carnegie Mellon University, August 2007.
Published as Technical Report CMU-CS-07-129.

Technical reports

Jana Dunfield and Frank Pfenning. Tridirectional typechecking.
Carnegie Mellon University technical report CMU-CS-04-117, 2004. (Extended version of POPL '04 paper.)

Jana Dunfield. Combining two forms of type refinements.
Carnegie Mellon University technical report CMU-CS-02-182, 2002.

Under submission

Jana Dunfield and Neel Krishnaswami. Bidirectional typing.
Submitted to ACM Computing Surveys, August 2019.

Drafts

Matthew A. Hammer, Jana Dunfield, Kyle Headley, Dimitrios J. Economou and Monal Narasimhamurthy.
Fungi: Typed incremental computation with names. arXiv:`1808.07826 [cs.PL]`.
Draft, July 2018.

Selected talks

Uncertainty is hope: towards a unified foundation of gradual typing (Logic and Semantics Seminar, University of Cambridge, April 2019).
Uncertainty is Hope: logic and gradual typing (invited tutorial, MFPS '18).
Gradual typing for refinements (Queen's University, Kingston, May 2017; Brock University, St. Catharines, June 2017).
Elaborating evaluation-order polymorphism (ICFP '15).
Complete and easy bidirectional typechecking for higher-rank polymorphism (ICFP '13).
Elaborating intersection and union types (ICFP '12).
Verifying functional programs with type refinements (IMDEA Software Institute, Madrid, Feb. 2010; Max Planck Institute for Software Systems, Saarbrücken, Mar. 2010).
Greedy bidirectional polymorphism (ML Workshop '09).
Programming with proofs and explicit contexts (PPDP '08).
Case analysis of higher-order data (LFMTP '08).
Refined typechecking with Stardust (PLPV '07).
Tridirectional typechecking (POPL '04).
Type assignment for unions and intersections in call-by-value languages (FoSSaCS '03).

Supervision **Graduate:**

- PhD supervisor
  - Dimitrios Economou (2019–)
- PhD supervisory committee (School of Computing, Queen's University)
  - Lama Moukahal (2018–)
  - Majid Babaei (2018–)
- PhD examinations and defences (School of Computing, Queen's University)
  - Benjamin Cecchetto (head's representative, defence, 2019)
  - Marwa Elsayed (head's representative, defence, 2018)
  - Heng Li (examiner, comprehensive examination, 2018)
  - Mojtaba Bagherzadeh (examiner, comprehensive examination, 2017)
  - Mohamed Sami Rakha (internal examiner, defence, 2017)
- PhD RPE (Research Proficiency Evaluation) committee member (Computer Science, University of British Columbia)
  - Joey Eremondi (2017)
  - Felipe Bañados Schwerter (2015)

Undergraduate:

- Quinn Pollock: CISC 499 project supervisor, January–April 2020 (School of Computing, Queen's University)
- Haadia Mufti: Summer research supervisor, May–July 2019 (School of Computing, Queen's University)
- Sean Remedios, Taylor Simpson, Dankai Shao, Yuhang Cao, Yuqi Wang: CISC 498 project supervisor, September 2018–April 2019 (School of Computing, Queen's University)
- Aniqah Mair: CISC 499 project supervisor, January–April 2019 (School of Computing, Queen's University)
- Elmar Tirtarahardja: CISC 499 project supervisor, January–April 2019 (School of Computing, Queen's University)
- Vincent Hui: CPSC 448 Directed Studies supervisor, January–May 2017 (Computer Science, University of British Columbia)

Service **Reviewing:**

- **Programme committee member**:
  - IC 2019
  - TFP 2019
  - ITRS 2010, 2016, 2018
  - ESOP 2016
  - ICFP 2016
  - PPDP 2014
  - MLW 2011
- **External review committee member**: ICFP 2018.
- **Journal reviewer:** *Information and Computation*; *ACM Computing Surveys*; *Computer Languages, Systems & Structures*; *ACM Transactions on Programming Languages and Systems*
- **Conference reviewer:** CSL, ESOP, ICFP, LICS, PLDI, POPL, SAC.
- **Workshop reviewer:** CPP, LFMTP, PLPV, RTA, WFLP.

Teaching CISC 865 Semantics of Programming Languages, Queen's University, Winter 2020, Winter 2019, Winter 2018 (enrolment **6**, **6**, **7**).

Graduate course, in Winter 2020 combined with undergraduate CISC 465 (enrolment **12**).

Instructor. Lectured; designed new course, created content and lecture notes; created and marked assignments; held office hours.

CISC 204 Logic for Computing Science, Queen's University, Fall 2019 (September–December 2019).

Enrolment at end of term: **350**.

Instructor. Lectured; created course content and lecture notes; created exams; supervised 12 teaching assistants; held office hours.

CISC 360 Programming Paradigms, Queen's University, Fall 2019 (September–December 2019).

Required for all Computing undergraduates. Enrolment at end of term: **57**.

Instructor. Lectured; revised CISC 260 course content and lecture notes; created assignments and exams; supervised two teaching assistants; held office hours.

CISC 260 Programming Paradigms, Queen's University, Fall 2018 (September–December 2018).

Required for all Computing undergraduates. Enrolment at end of term: **73**.

Instructor. Lectured; created course content and lecture notes; created assignments and exams; marked exams; supervised two teaching assistants; held office hours.

CISC 327 Software Quality Assurance, Queen's University, Fall 2018 (enrolment at end of term: **120**) and Fall 2017 (enrolment at end of term: **140**). Required for Software Design undergraduates.

Instructor. Lectured; updated course content and lecture notes; created assignments and exams; marked exams; supervised four teaching assistants; held office hours.

CPSC 311 Definition of Programming Languages, University of British Columbia, 2016 Winter 1 (September–December 2016).

Undergraduate elective. Enrolment at end of term: **119**.

Sessional Lecturer. Lectured; updated course content and wrote lecture notes; created assignments and exams; marked portions of exams; supervised five teaching assistants; held office hours.

CPSC 311 Definition of Programming Languages, University of British Columbia, 2015 Winter 1 (September–December 2015).

Undergraduate elective. Enrolment at end of term: **108**.

Sessional Lecturer. Lectured; updated course content and wrote lecture notes; created assignments and exams; marked portions of exams; supervised three teaching assistants; held office hours.

COMP 302 Programming Languages and Paradigms, McGill University, Winter 2010. Required for undergraduates in CS and several other majors; typically taken in U1 or U2 (Québec system; roughly, 2nd and 3rd year). Enrolment at end of term: 61. **Course lecturer.** Lectured, wrote and revised lecture notes, created assignments and exams, marked portions of assignments and exams, supervised teaching assistants, and held office hours. Student evaluations (1–5 scale):

"Overall, this is an excellent course"	mean 4.0
"Overall, this instructor is an excellent teacher"	mean 4.2
union of all questions	mean 4.2

COMP 523 Language-Based Security, McGill University, Winter 2008. Master's-level graduate course. Teaching assistant for Brigitte Pientka. Marked assignments and held office hours. (Ended early due to a teaching assistant strike.)

Teaching, continued

15–312 Foundations of Programming Languages, Carnegie Mellon University, Fall 2002. Satisfies area requirement for undergraduate CS majors. Teaching assistant for Frank Pfenning. Held weekly recitations, guest lectured, prepared and graded assignments, assisted in exam creation and grading, and held office hours. Student evaluations (1–4 scale):

	min.	mean	max.
"Overall effectiveness"	3	3.73	4
union of all questions	3	3.68	4

15–212 Principles of Programming, Carnegie Mellon University, Spring 2001. Required for undergraduate CS majors; typically taken in the second year. Teaching assistant for Michael Erdmann and Jeannette Wing. Held weekly recitation, prepared and graded assignments, assisted with exam creation and grading, and held office hours.