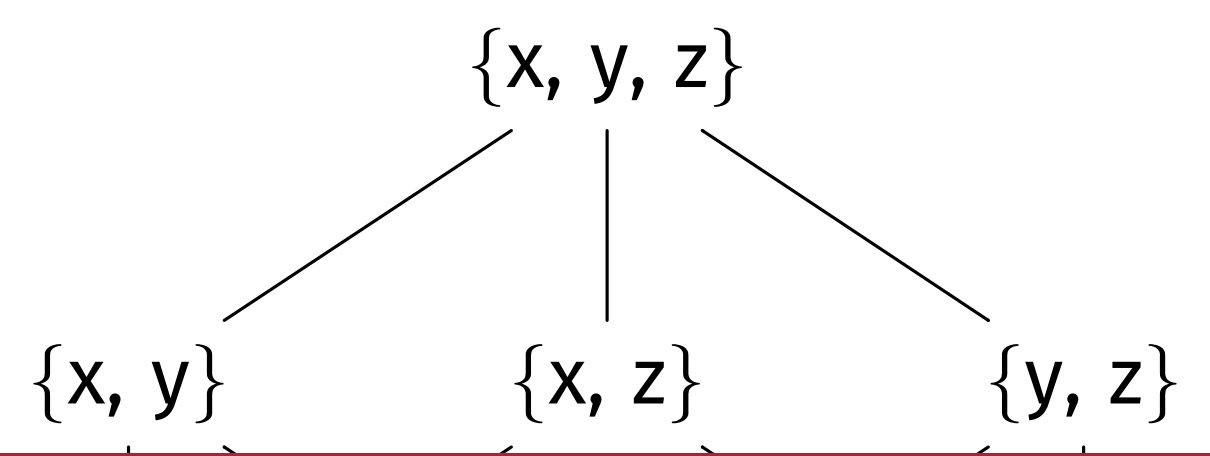


Abstract Interpretation

Software Engineering
Programming Languages

universität ulm

The Why

```
public static void main(String[] args) {
    int a = 1;
    double r = Math.random() * 10;
    if (r > 5) {
       a = 2;
    }
    System.out.println(a);
}
```

The Why

```
public static void main(String[] args) {
    int a = 1;
    double r = Math.random() * 10;
    if (r > 5) {
       a = 2;
    }
    System.out.println(a);
}
```

The Why

```
public static void main(String[] args) {
    int a = 1;
    double r = Math.random() * 10;
    if (r > 5) {
       a = 2;
    }
    System.out.println(a);
}
```

$\{a \in \{1\}\}$

The Why

```
public static void main(String[] args) {
    int a = 1;                          ⦃ a ∈ {1} ⦄
    double r = Math.random() * 10;      ⦃ r ∈ [0..10) ⦄
    if (r > 5) {
       a = 2;
    }
    System.out.println(a);
}
```

The Why

```
public static void main(String[] args) {
    int a = 1;                          ⟅ a ∈ {1} ⟆
    double r = Math.random() * 10;      ⟅ r ∈ [0..10) ⟆
    if (r > 5) {
       a = 2;                           ⟅ a ∈ {2} ⟆
    }
    System.out.println(a);
}
```

The Why

```
public static void main(String[] args) {
    int a = 1;                          ⦃ a ∈ {1} ⦄
    double r = Math.random() * 10;      ⦃ r ∈ [0..10) ⦄
    if (r > 5) {
       a = 2;                           ⦃ a ∈ {2} ⦄
    }
    System.out.println(a);              ⦃ a ∈ {1,2} ⦄
}
```

The Why

```
public static void main(String[] args) {
    int a = 1;                          ⟅ a ∈ {1} ⟆
    double r = Math.random() * 10;      ⟅ r ∈ [0..10) ⟆
    if (r > 5) {
       a = 2;                           ⟅ a ∈ {2} ⟆
    }
    System.out.println(a);              ⟅ a ∈ {1,2} ⟆ → Valid? Ok? Safe?
}
```

The Why

```
public static void main(String[] args) {
    int a = 1;
    double r = Math.random() * 10;
    if (r > 5) {
       a = 2;
    }
    System.out.println(a);
}
```

$⟅ a \in \{1\} ⟆$

$⟅ r \in [0..10) ⟆$

$⟅ a \in \{2\} ⟆$

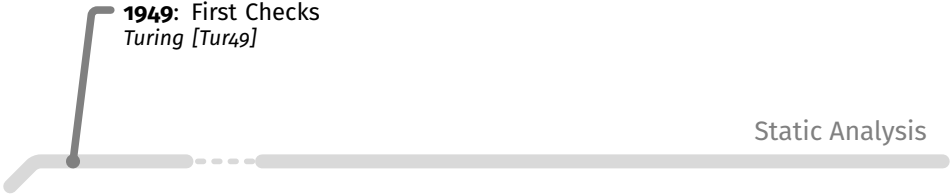
$⟅ a \in \{1, 2\} ⟆$ → Valid? Ok? Safe?

- We want to proof, that a program satisfies certain properties

Origins

Based on the amazing “Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation” by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins

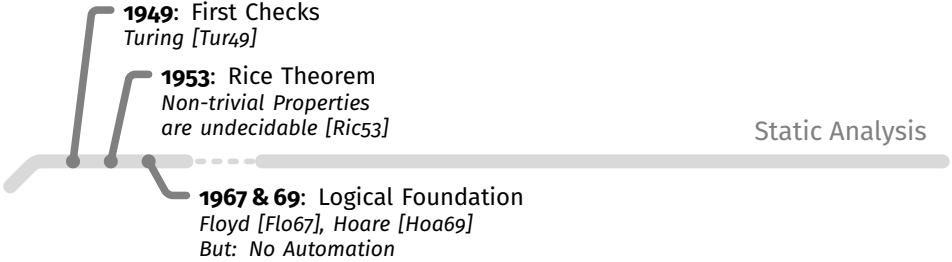


Based on the amazing “Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation” by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins

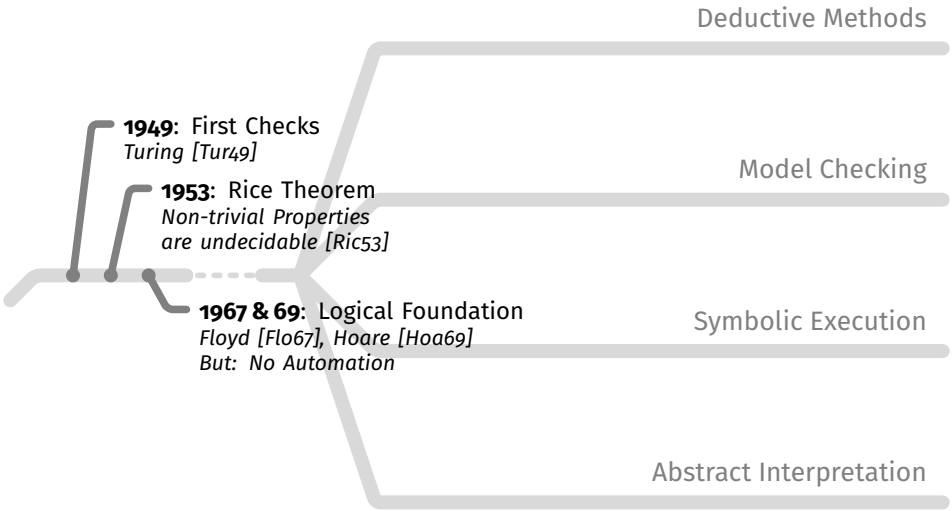
Based on the amazing "Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation" by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins



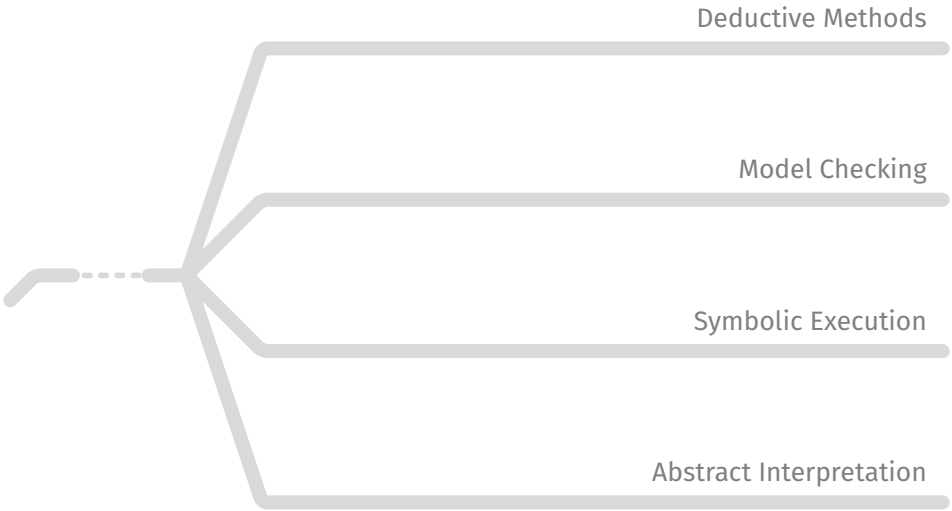
Based on the amazing "Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation" by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins



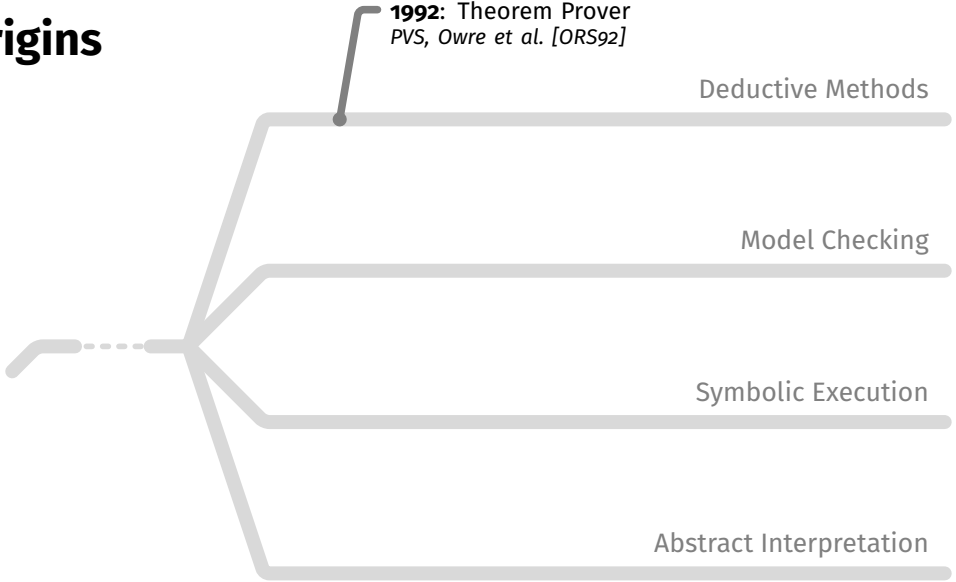
Based on the amazing "Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation" by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins



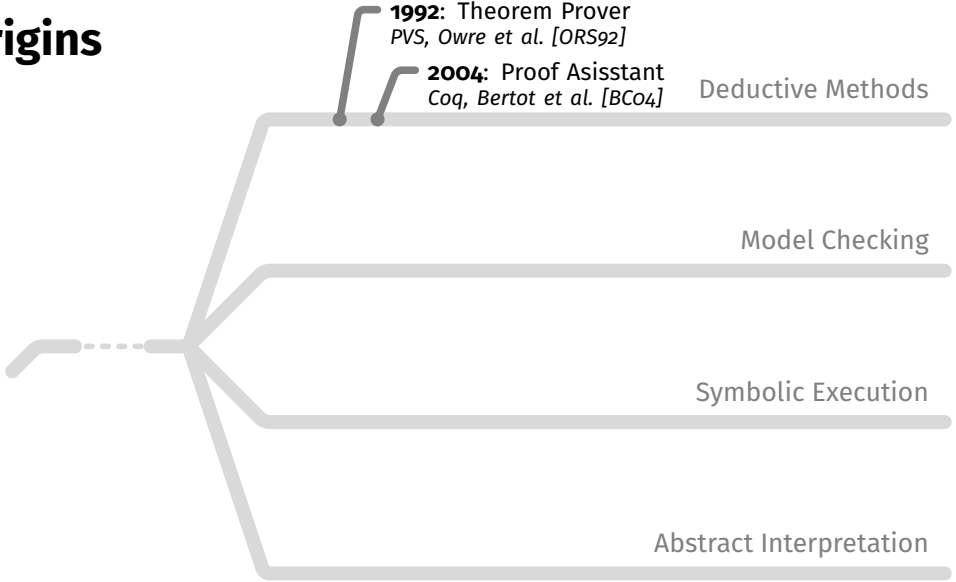
Based on the amazing “Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation” by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins



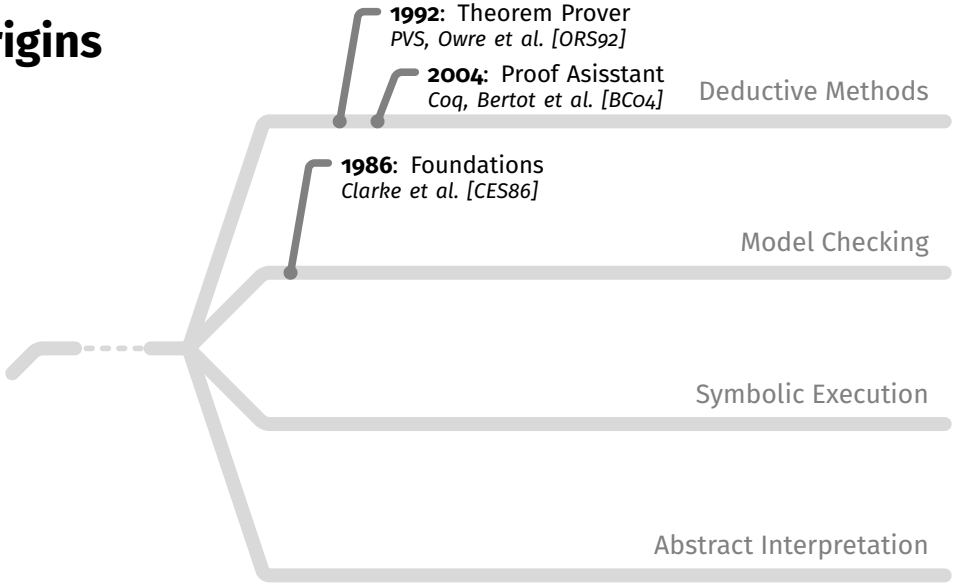
Based on the amazing "Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation" by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins



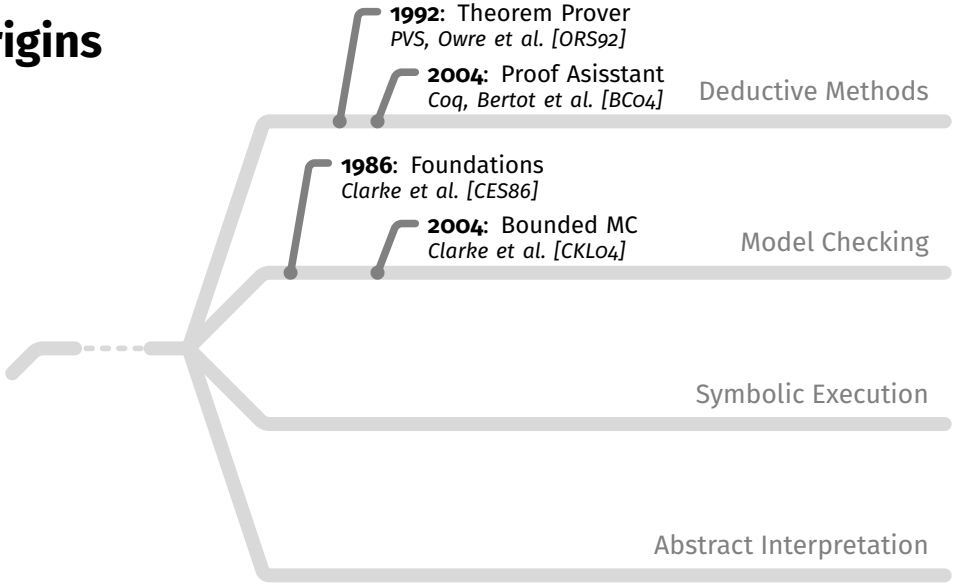
Based on the amazing "Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation" by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins



Based on the amazing "Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation" by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins



Based on the amazing "Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation" by Miné [Min17] and https://www.di.ens.fr/ cousot/AI/

Origins

Content of first slide [TODO: timeline]

What is a Property? Set basis Poset etc. I have to abstract! Galois, Semantics Principles of Abstract Interpretation book

References I

[BC04] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. ISBN: 978-3-642-05880-6. DOI: 10.1007/978-3-662-07964-5. URL: https://doi.org/10.1007/978-3-662-07964-5.

[CES86] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications". In: *ACM Trans. Program. Lang. Syst.* 8.2 (1986), pp. 244–263. DOI: 10.1145/5397.5399. URL: https://doi.org/10.1145/5397.5399.

[CKL04] Edmund Clarke, Daniel Kroening, and Flavio Lerda. "A tool for checking ANSI-C programs". In: *Tools and Algorithms for the Construction and Analysis of Systems: 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29-April 2, 2004. Proceedings 10*. Springer. 2004, pp. 168–176.

References II

[Flo67] Robert W. Floyd. “Assigning Meanings to Programs”. In: *Proc. of the American Mathematical Society Symposia on Applied Mathematics*. Vol. 19. 1967, pp. 19–32.

[Hoa69] C. A. R. Hoare. “An Axiomatic Basis for Computer Programming”. In: *Commun. ACM* 12.10 (1969), pp. 576–580. DOI: 10.1145/363235.363259. URL: https://doi.org/10.1145/363235.363259.

[Min17] Antoine Miné. “Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation”. In: *Found. Trends Program. Lang.* 4.3-4 (2017), pp. 120–372. DOI: 10.1561/2500000034. URL: https://doi.org/10.1561/2500000034.

[ORS92] Sam Owre, John M. Rushby, and Natarajan Shankar. “PVS: A Prototype Verification System”. In: *Automated Deduction - CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings*. Ed. by Deepak Kapur. Vol. 607. Lecture Notes in Computer Science. Springer, 1992, pp. 748–752. DOI: 10.1007/3-540-55602-8_217. URL: https://doi.org/10.1007/3-540-55602-8%5C_217.

References III

[Ric53] Henry Gordon Rice. “Classes of recursively enumerable sets and their decision problems”. In: *Transactions of the American Mathematical society* 74.2 (1953), pp. 358–366.

[Tur49] Alan Turing. “Checking a large routine”. In: *Report of a Conference on High Speed Automatic Calculating Machines*. 1949, pp. 67–69.