Negative Perceptions About the Applicability of Source-to-Source Compilers in HPC

06.12.2022

Mariusz Tkocz

Seminar Supercomputer: Forschung und Innovation
Arbeitsbereich Wissenschaftliches Rechnen, Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Negative Eindrücke über die Anwendbarkeit von Source-to-Source Compilern im Hochleistungsrechnen

Gliederung

- Vorstellung Paper
- Source-to-Source Compiler
- Methodik der Analyse
- Ergebnisse und Lösungsvorschläge
- Verwandte Arbeit
- Schlussfolgerung

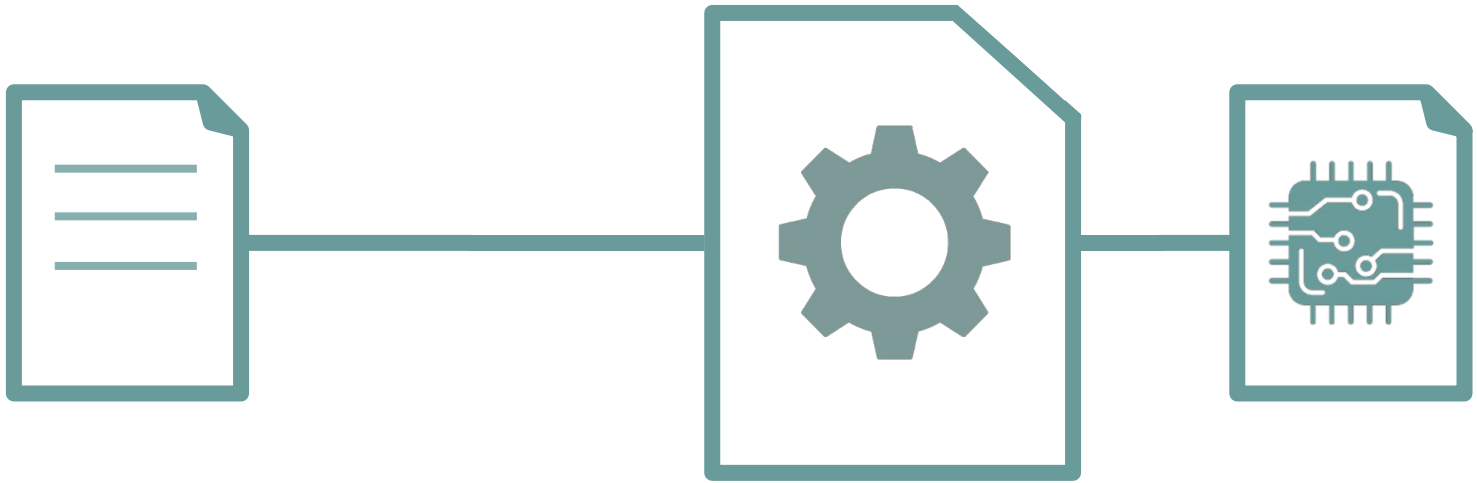
Vorstellung Paper

Vorstellung Paper

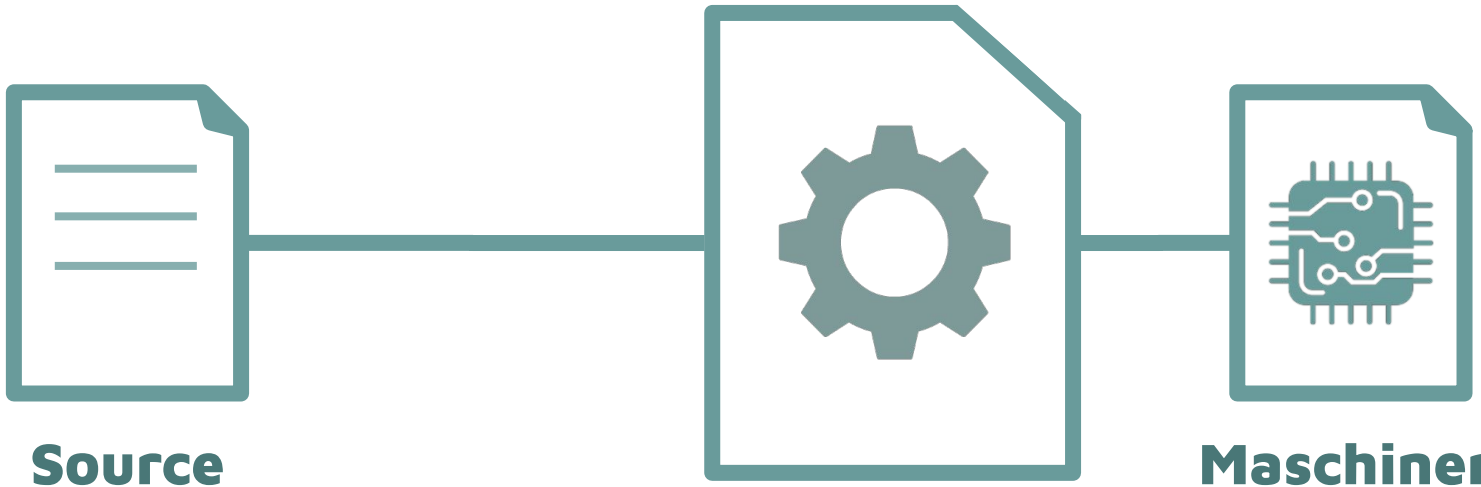
- 13. November 2021
- “Negative Perceptions About the Applicability of Source-to-Source Compilers in HPC: A Literature Review”
- Reed Milewicz et al. [5]
- Sandia National Laboratories, Albuquerque, USA
- U.S. Department of Energy by Lawrence Livermore National Laboratory

Source-To-Source Compiler

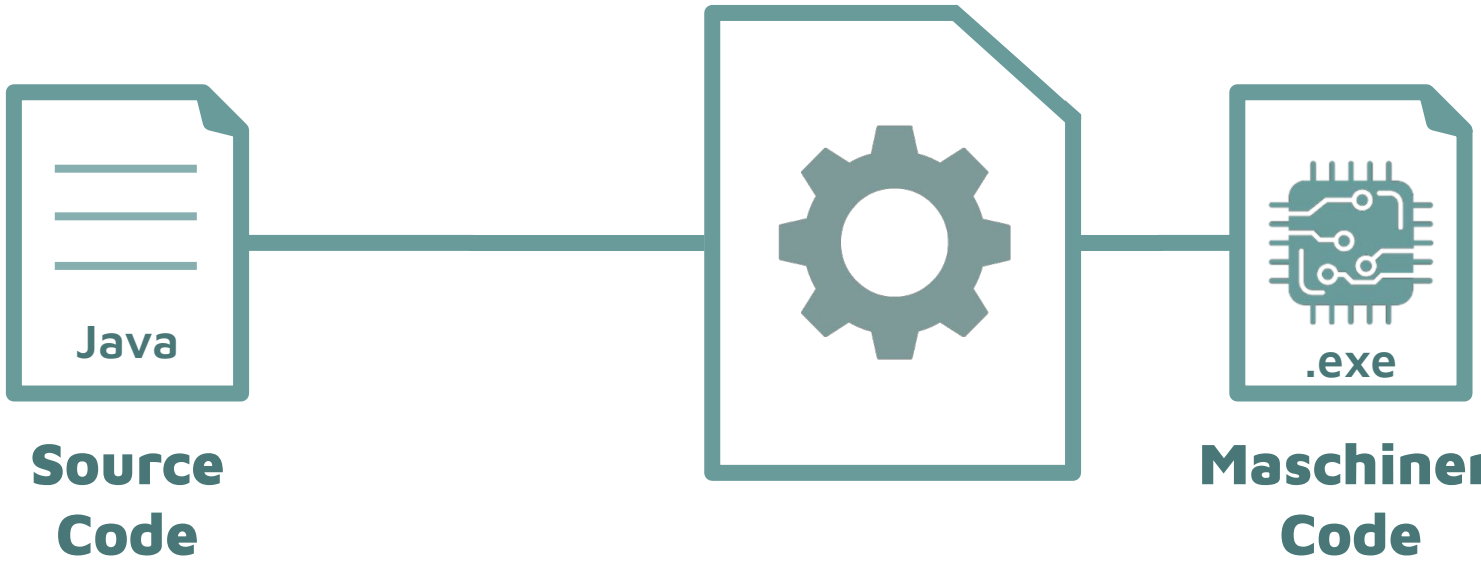
Compiler



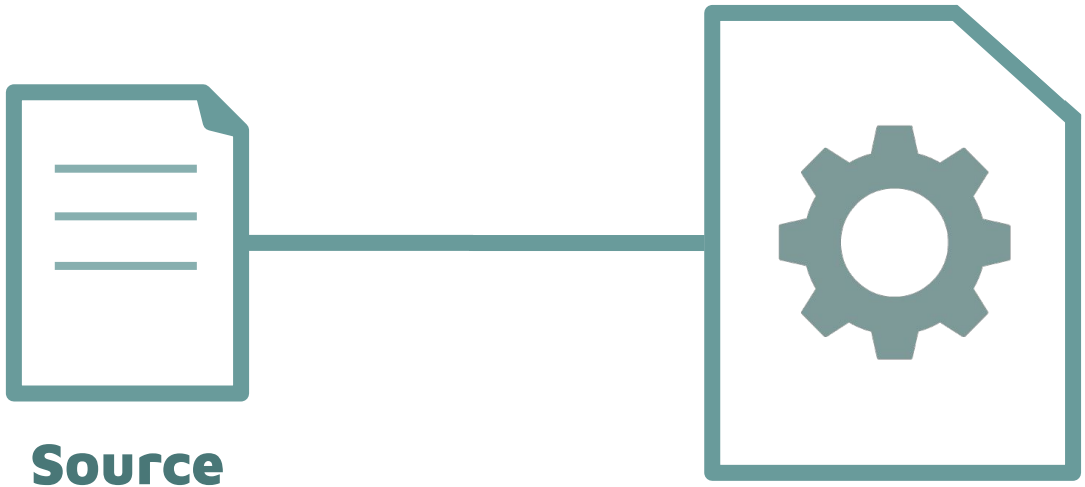
Compiler



Compiler

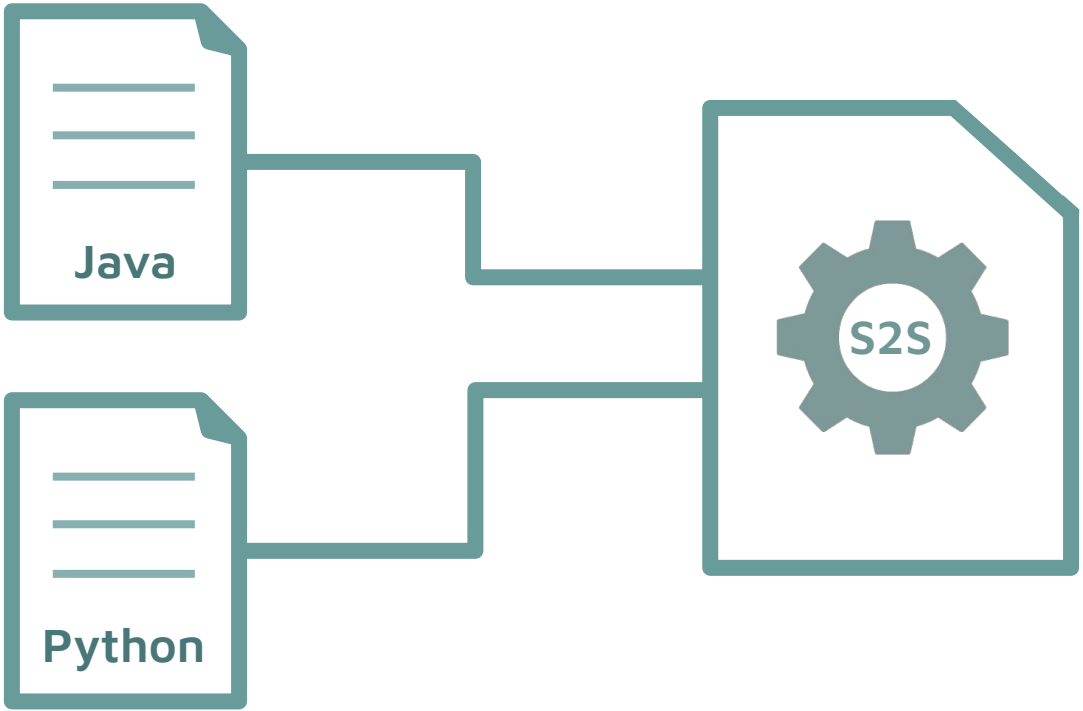


Compiler



Source-To-Source Compiler

Source-To-Source Compiler

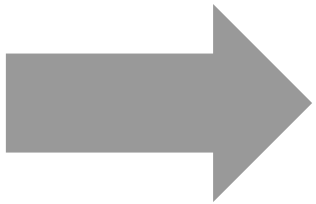


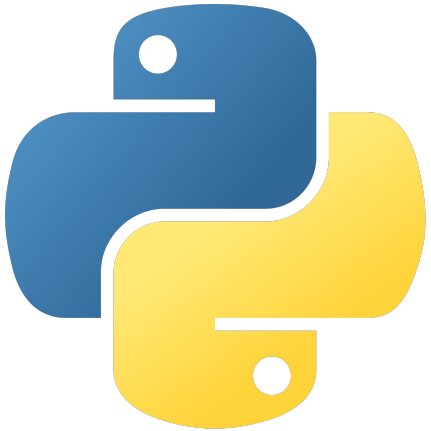
Source-To-Source Compiler

Source-To-Source Compiler

Anwendung

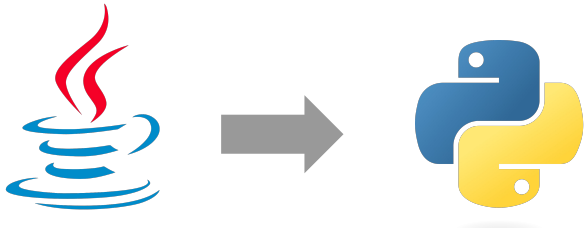
Anwendung





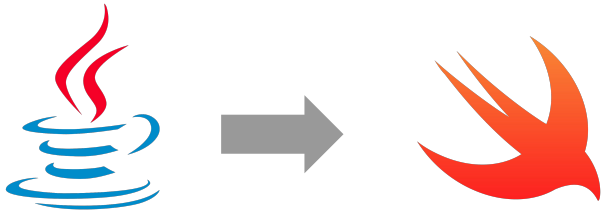
https://www.python.org/community/logos/
https://brandslogos.com/j/java-logo-1/

Anwendung



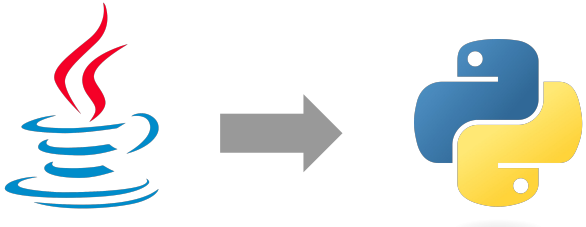
- Vielsprachige Entwicklung
- Portierung auf andere Plattformen
- Migration von Legacy Code
- Transparenz, Flexibilität und Erweiterbarkeit von Black Boxen

Anwendung



- Vielsprachige Entwicklung
- Portierung auf andere Plattformen
- Migration von Legacy Code
- Transparenz, Flexibilität und Erweiterbarkeit von Black Boxen

Anwendung

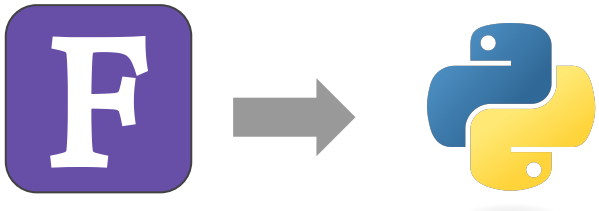


- Vielsprachige Entwicklung
- Portierung auf andere Plattformen
- Migration von Legacy Code
- Transparenz, Flexibilität und Erweiterbarkeit von Black Boxen

Anwendung

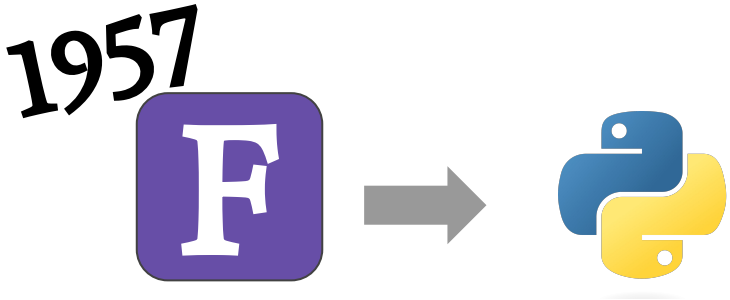
- Vielsprachige Entwicklung
- Portierung auf andere Plattformen
- Migration von Legacy Code
- Transparenz, Flexibilität und Erweiterbarkeit von Black Boxen

Anwendung



- Vielsprachige Entwicklung
- Portierung auf andere Plattformen
- Migration von Legacy Code
- Transparenz, Flexibilität und Erweiterbarkeit von Black Boxen

Anwendung



- Vielsprachige Entwicklung
- Portierung auf andere Plattformen
- Migration von Legacy Code
- Transparenz, Flexibilität und Erweiterbarkeit von Black Boxen

Anwendung

1957

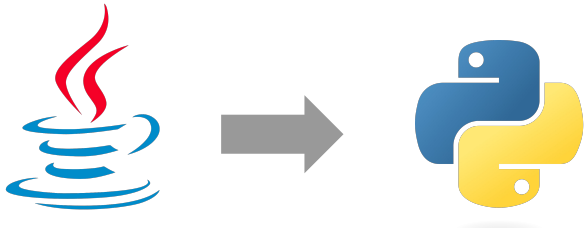
https://www.computerhistory.org/timeline/1960/
https://www.pinterest.de/pin/760615824541908213/

Anwendung

1957

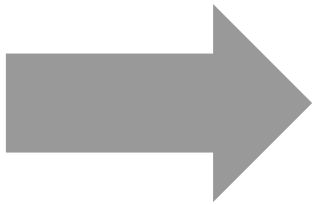
https://www.computerhistory.org/timeline/1960/
https://www.pinterest.de/pin/760615824541908213/

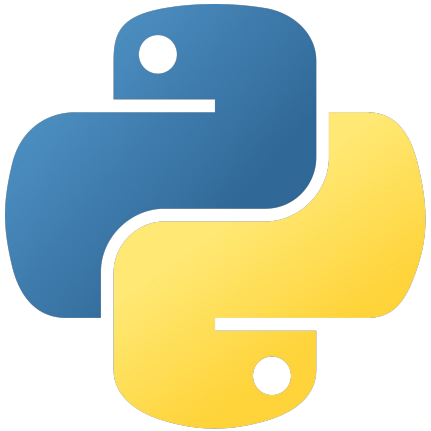
Anwendung



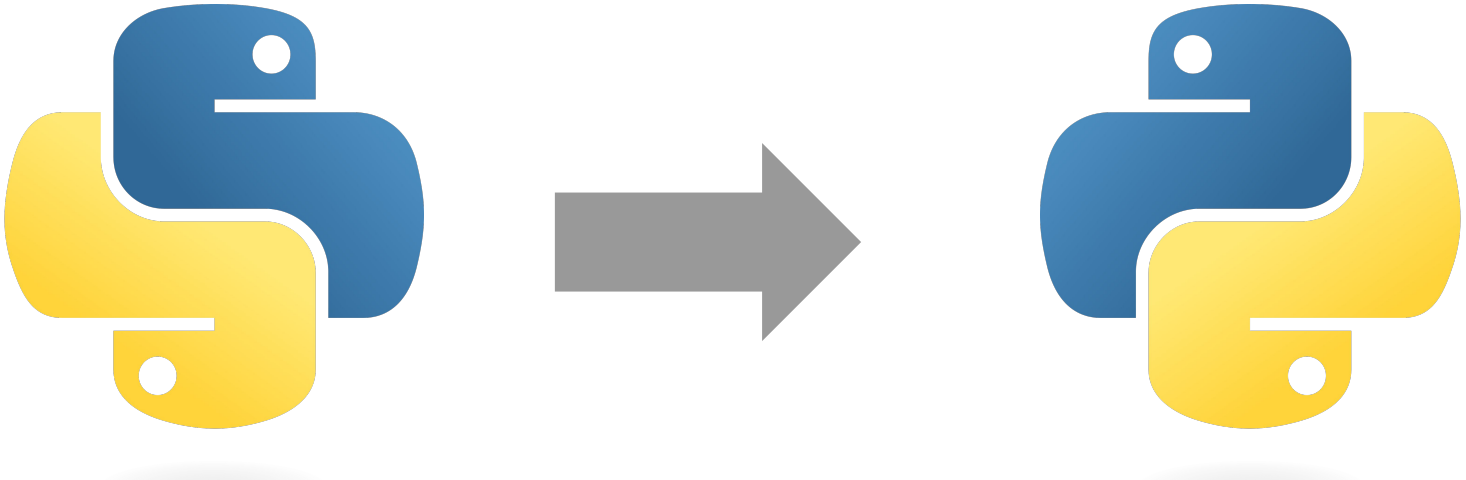
- Vielsprachige Entwicklung
- Portierung auf andere Plattformen
- Migration von Legacy Code
- Transparenz, Flexibilität und Erweiterbarkeit von Black Boxen

Anwendung

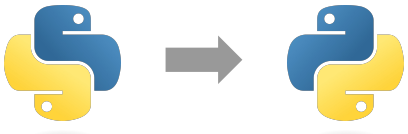




Anwendung

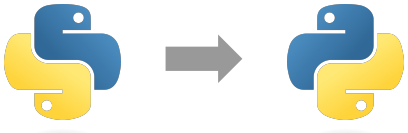


Anwendung



- Refactoring
- Fehlerüberprüfung
- Leistungsoptimierung
  - Schleifen Optimierung
  - Optimierung der Datenanordnung

Anwendung



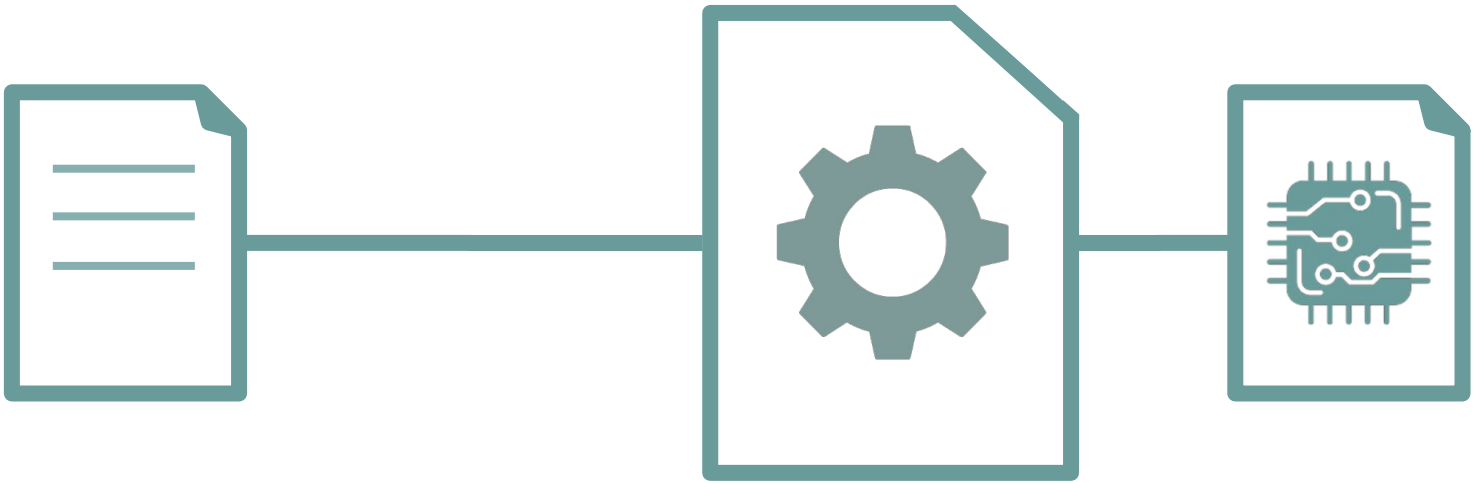
- Code Verjüngung
- Versionen Update (Python 2 → Python 3)
- Backward Compatibility (Python 3 → Python 2)

Anwendung HPC

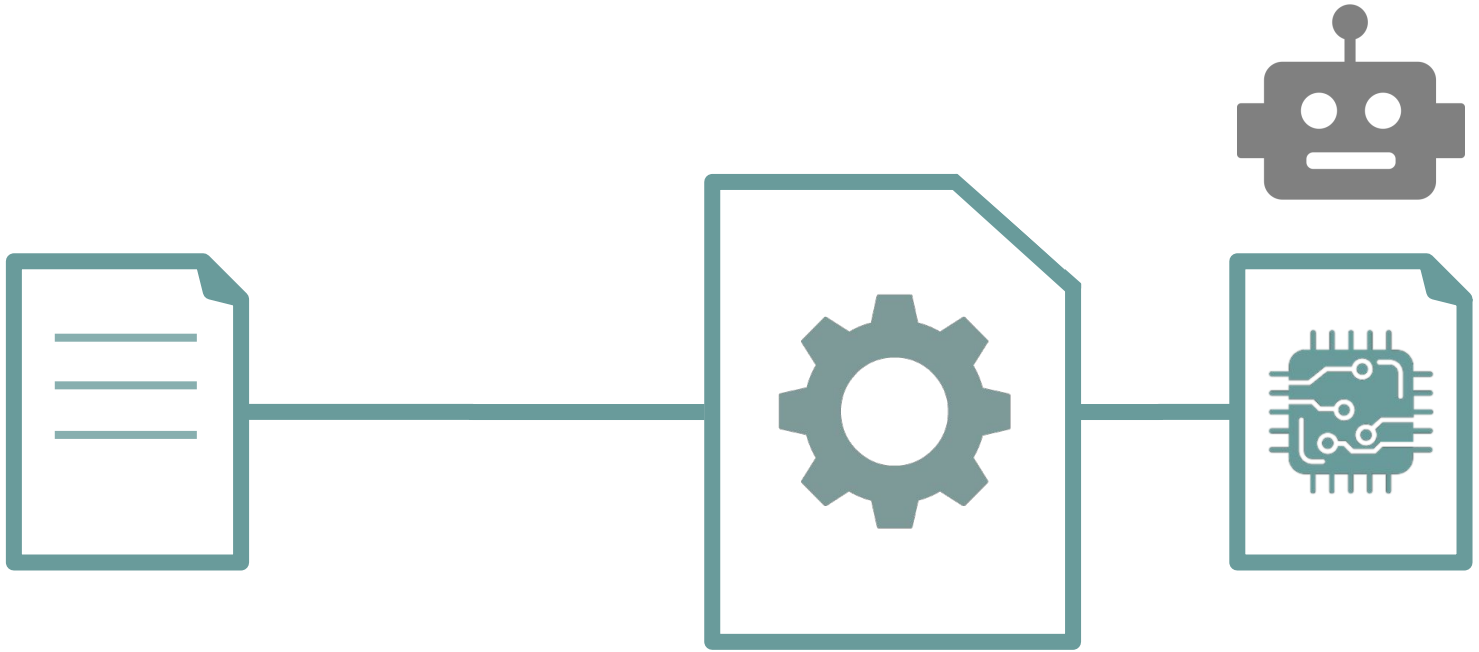
Anwendung HPC

- Entwurf domänespezifischer Sprachen für wiss. Rechnen
- Automatische Parallelisierung
- Fehlertolerantes Rechnen
- Instrumentierung für Checkpointing
- Automatisches Differenzieren
- Konvertieren zwischen Parallelisierungskonstrukten und Bibliotheken

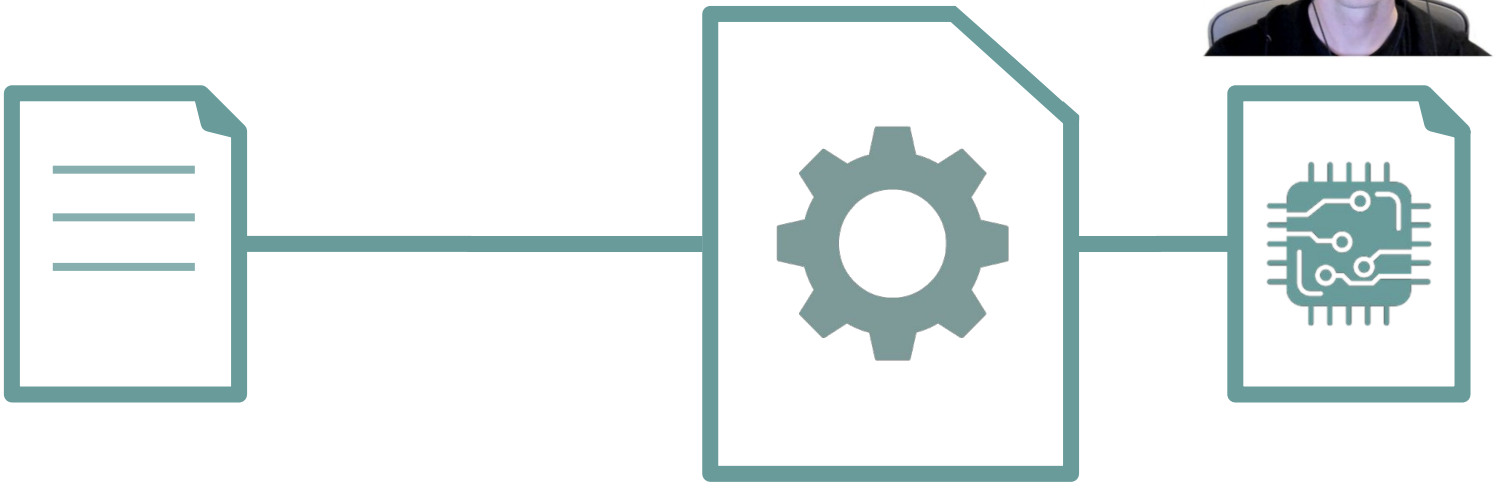
Anwendung



Anwendung

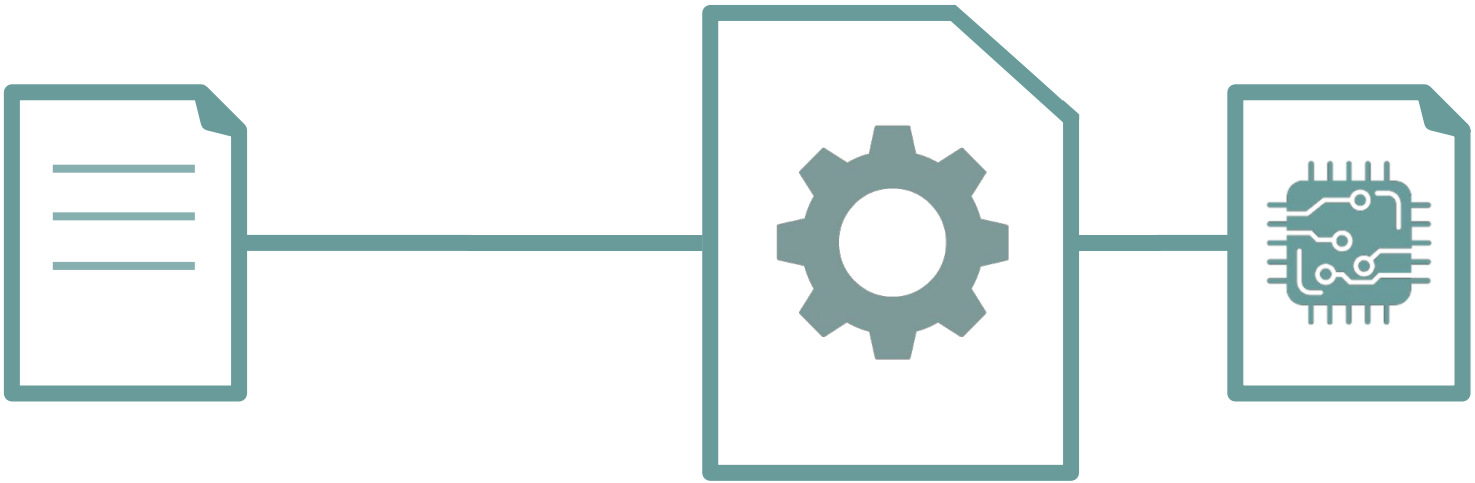


Anwendung



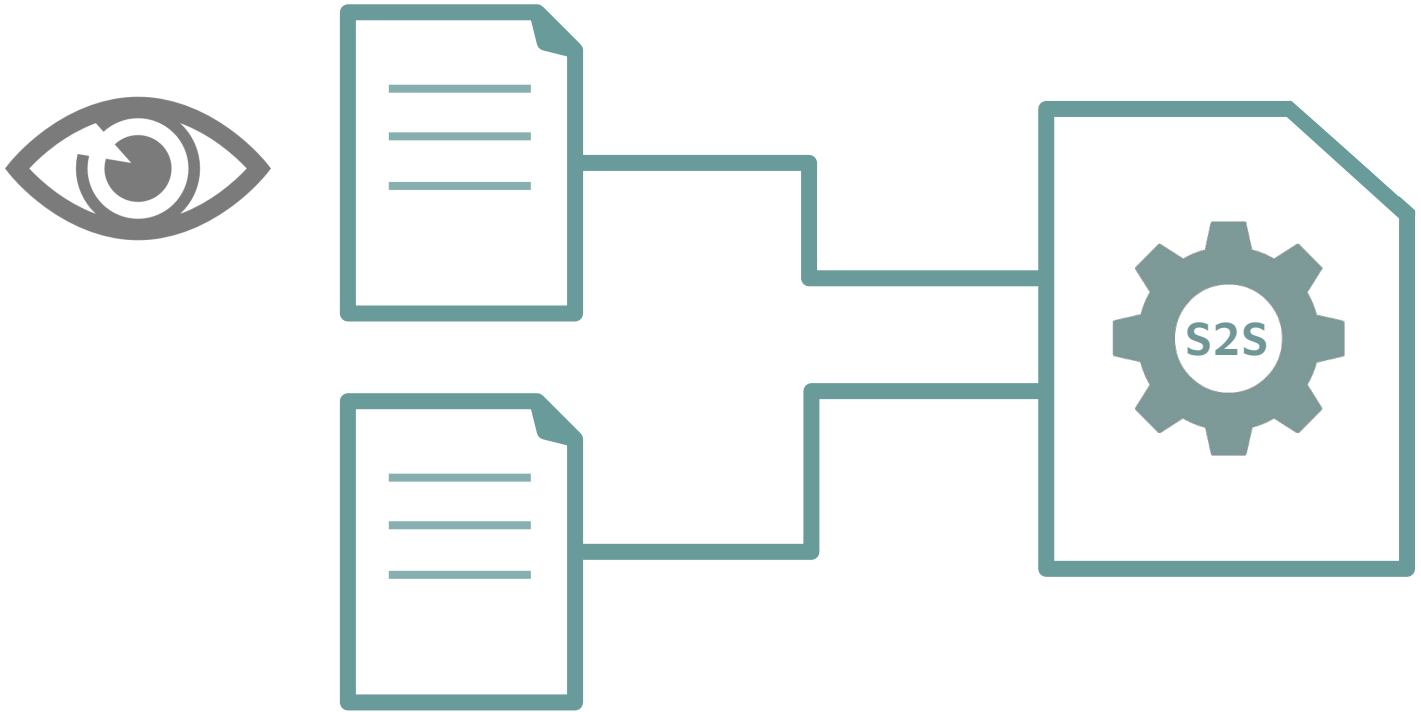
https://www.youtube.com/watch?v=hAsZCTL__lo

Anwendung



Anwendung

Anwendung



Methodik der Analyse

“ In welchem Kontext und aus welchen Gründen entscheiden sich HLR-Forscher gegen die Benutzung von S2S Compilern? ”

Methodik der Analyse

+

HARZING´S
PUBLISH
OR PERISH

https://github.com/FortAwesome/Font-Awesome/issues/17287

Methodik der Analyse

Query	Search string
Source-to-source terminology	(“source-to-source compiler” OR “s2s” OR “transcompiler” OR “transpiler”) AND (“high performance computing” OR “HPC”)
Popular source-to-source frameworks	(“compiler” AND (“ROSE” OR “Mercurium” OR “Insieme” OR “Clang” OR “OpenARC” OR “Xevolver” OR “Omni” OR “Pluto” OR “Clava”)) AND (“high performance computing” OR “HPC”)
Supplementary search	(“source-to-source” OR “s2s” OR “transcompiler” OR “transpiler”) AND (“weakness” OR “drawback” OR “limitation” OR “shortcoming”) AND (“high performance computing” OR “HPC”)

Negative Perceptions About the Applicability of S2S Compilers in HPC Table 1.

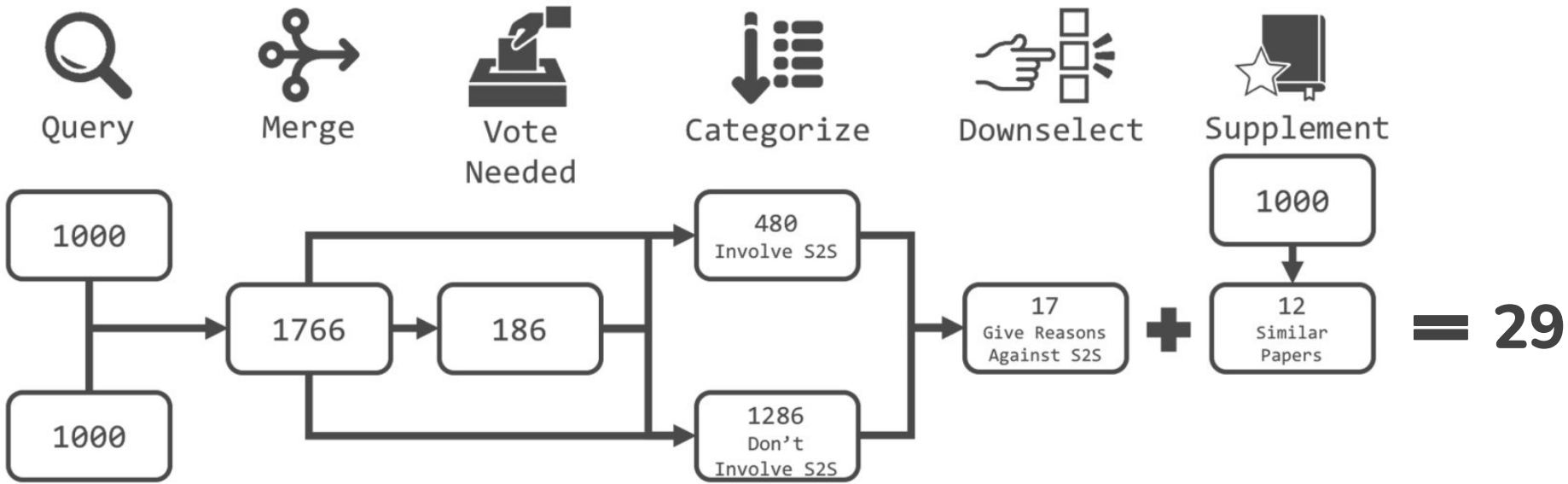
Methodik der Analyse

- 01 Januar 2014 - 31 Dezember 2019
- 2 * Suche Oktober 2020 = 1000
- 1 * Suche März 2021 = 1000
- 1000 + 1000 = 1766

Methodik der Analyse

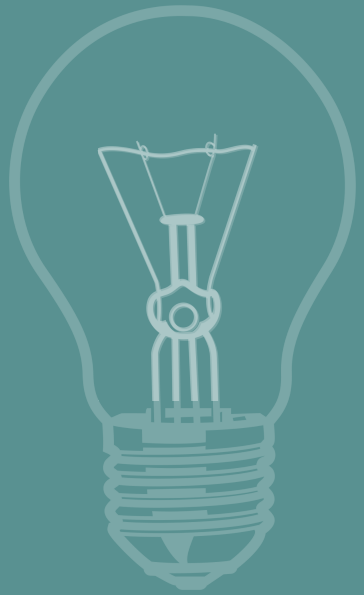
- Kein Fokus auf Hochleistungsrechnen
- Kein Verweis / keine Benutzung von S2S Compilern
- Kein Peer Review durchlaufen
- Einführungspaper für Bücher, Workshops etc.
- Gleiches Experiment, zwei Paper
- Nicht auf Englisch

Methodik der Analyse



Negative Perceptions About the Applicability of S2S Compilers in HPC Fig. 1.

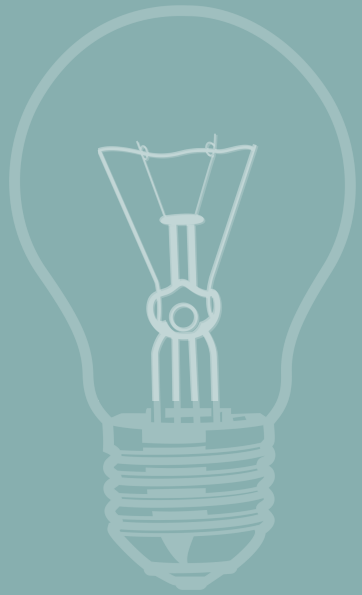
Ergebnisse und Lösungsvorschläge



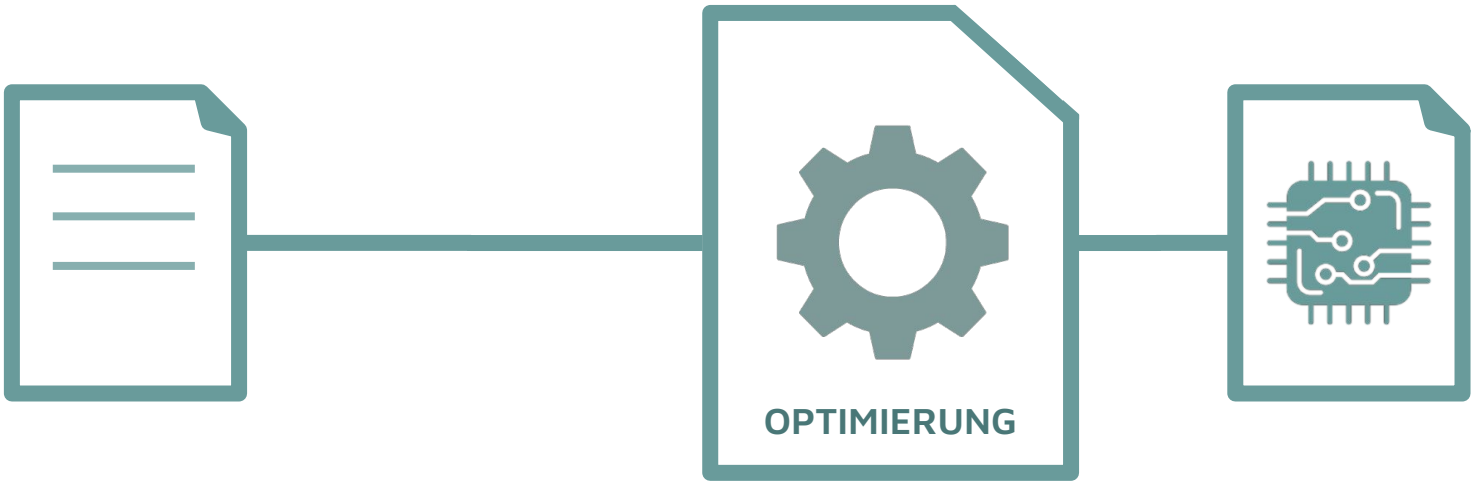
Ergebnisse

1. Verhindert Compiler Optimierungen
2. Schwer zu Erweitern
3. Fragile & Komplexe Arbeitsabläufe
4. Parsing sehr schwer
5. Lückenbüßer Lösung

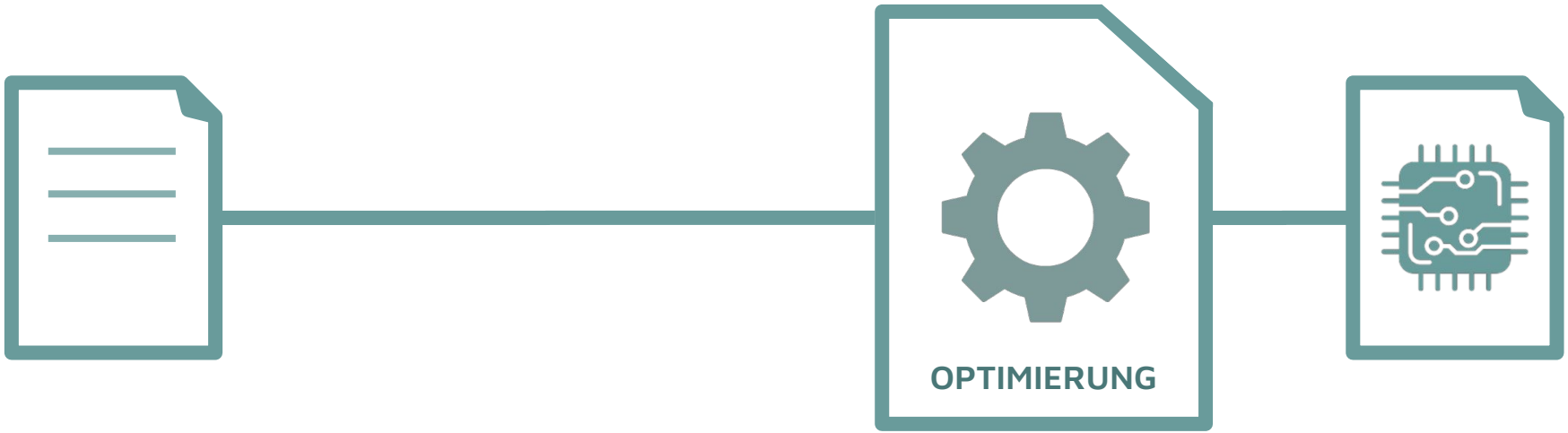
Verhindert Compiler Optimierung



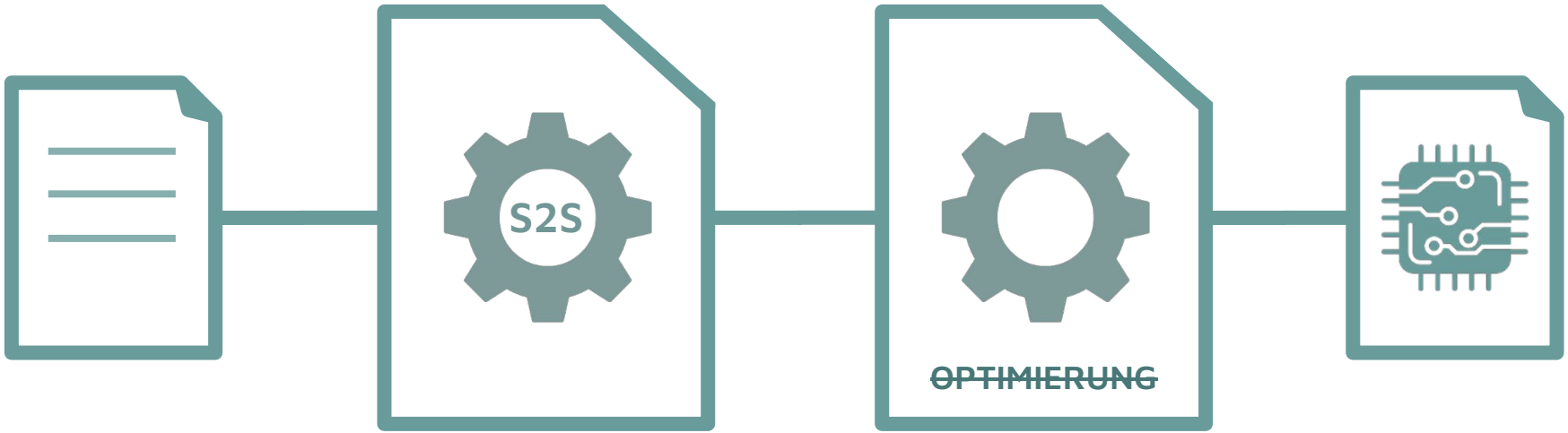
1. Verhindert Compiler Optimierungen



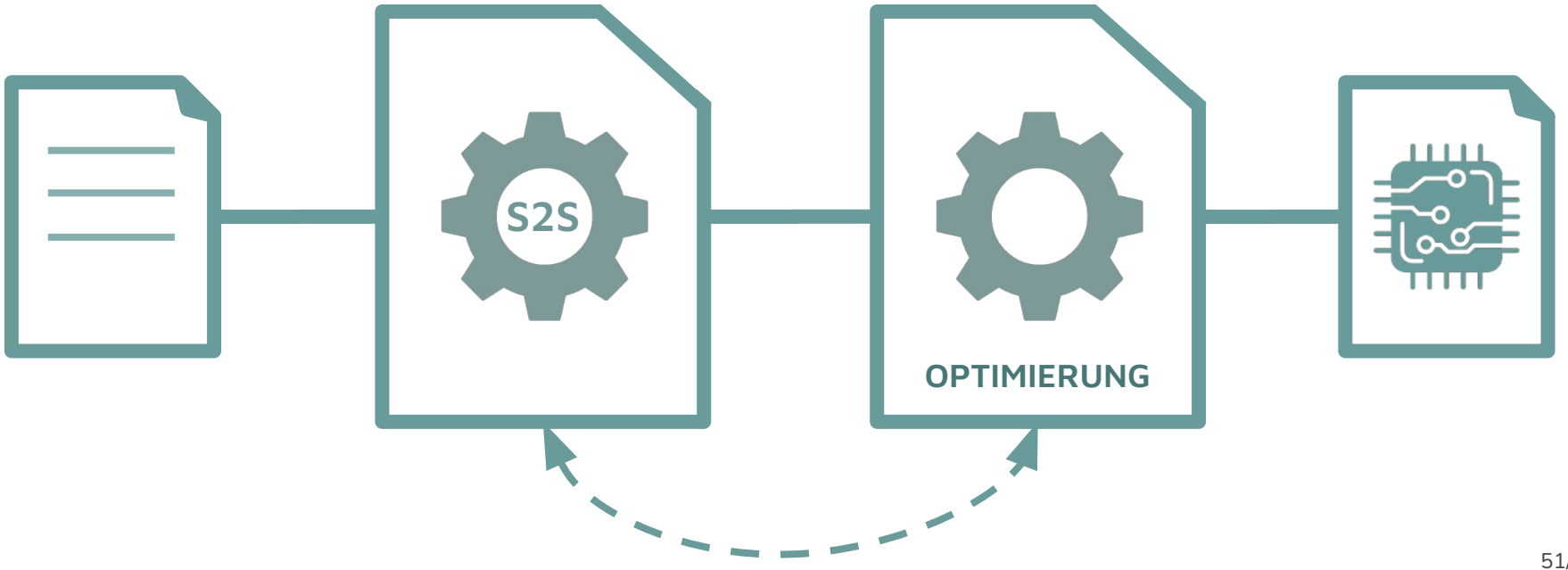
1. Verhindert Compiler Optimierungen



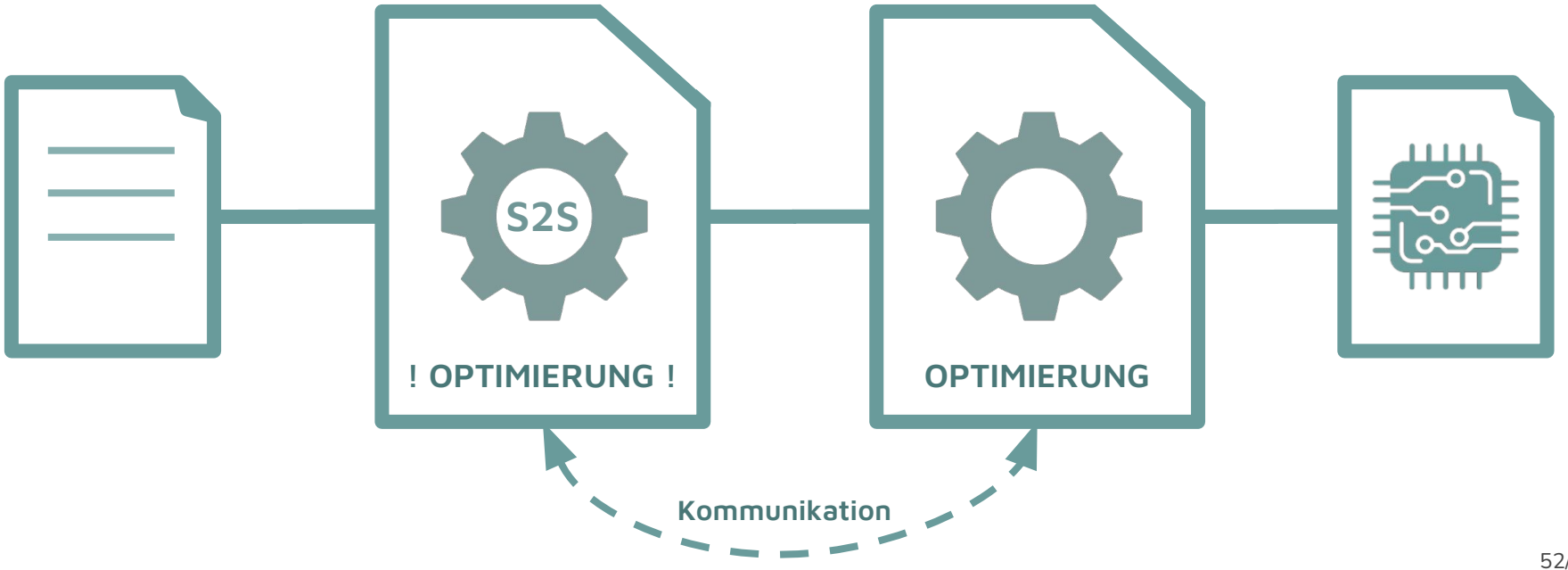
1. Verhindert Compiler Optimierungen



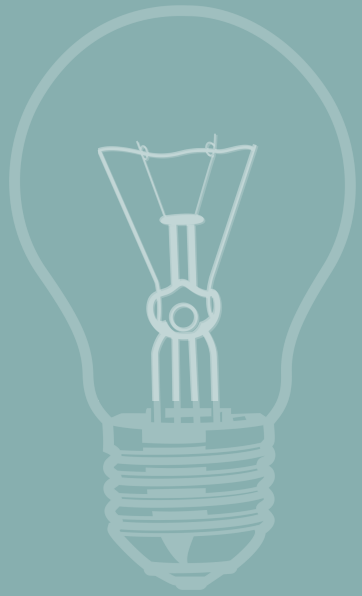
Re: Co-Design fördern



Re: Co-Design fördern



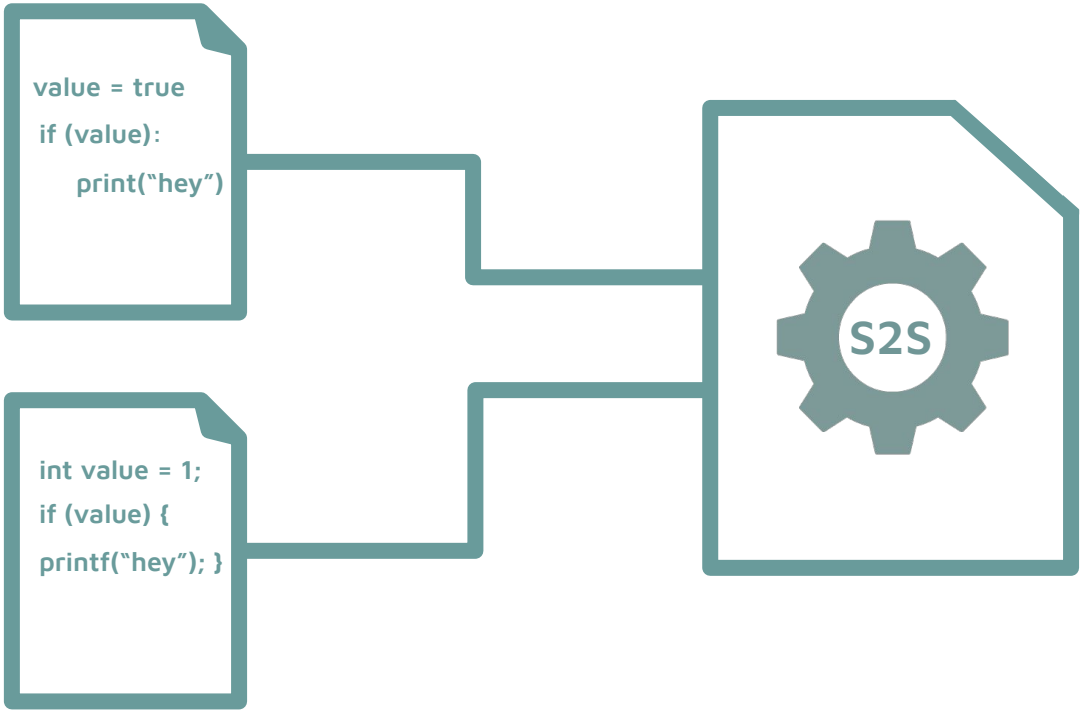
Schwer zu Erweitern



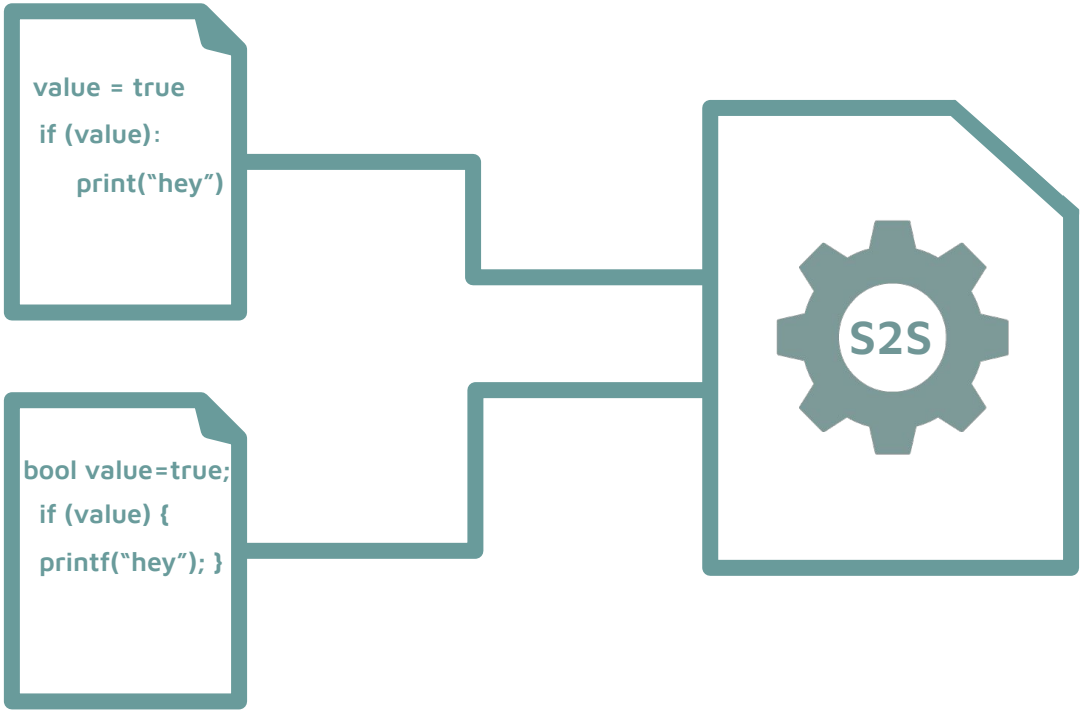
2. Schwer zu Erweitern

2. Schwer zu Erweitern

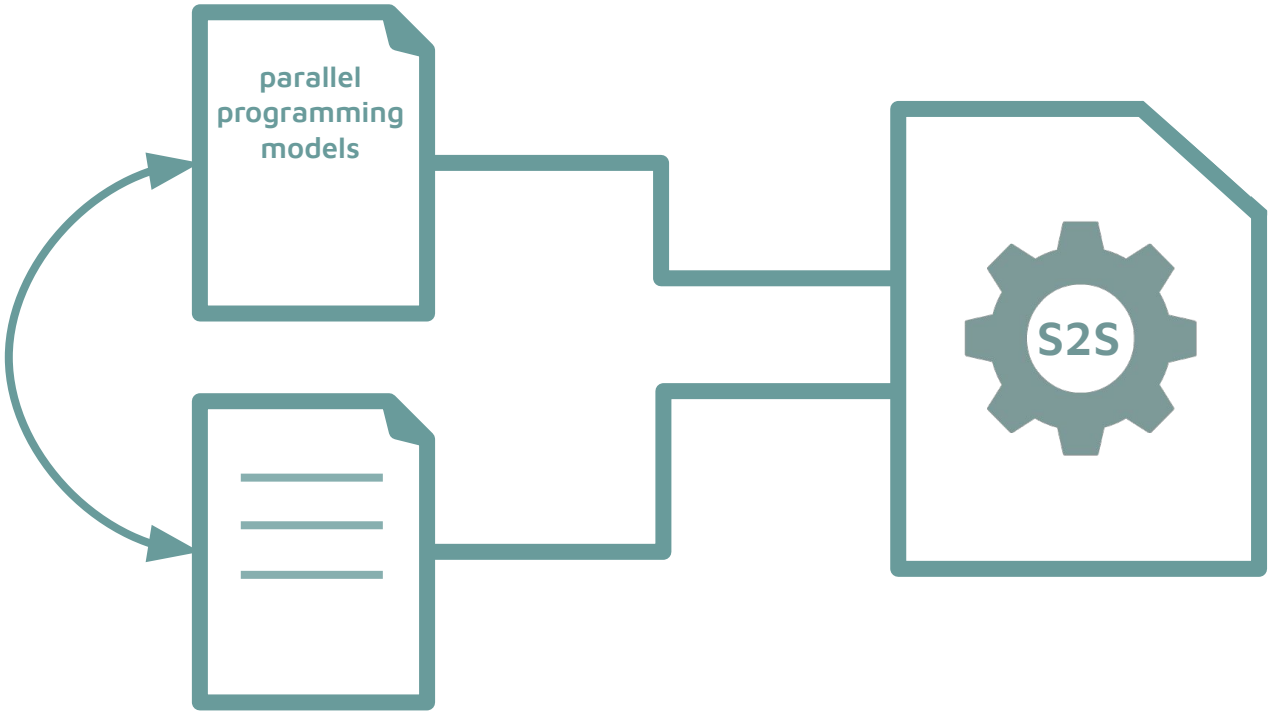
2. Schwer zu Erweitern



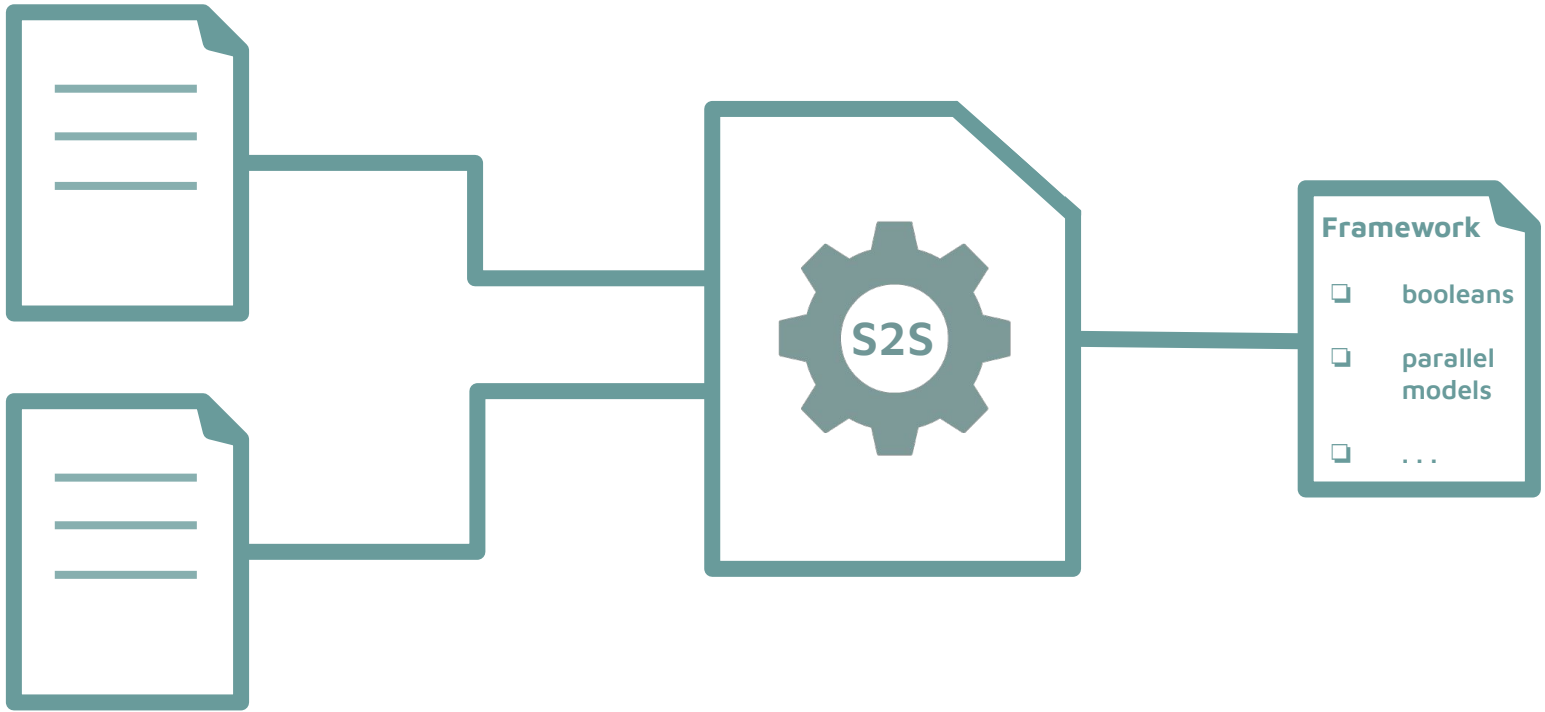
2. Schwer zu Erweitern



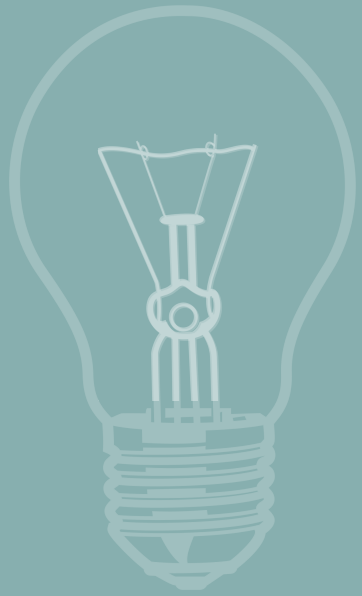
2. Schwer zu Erweitern



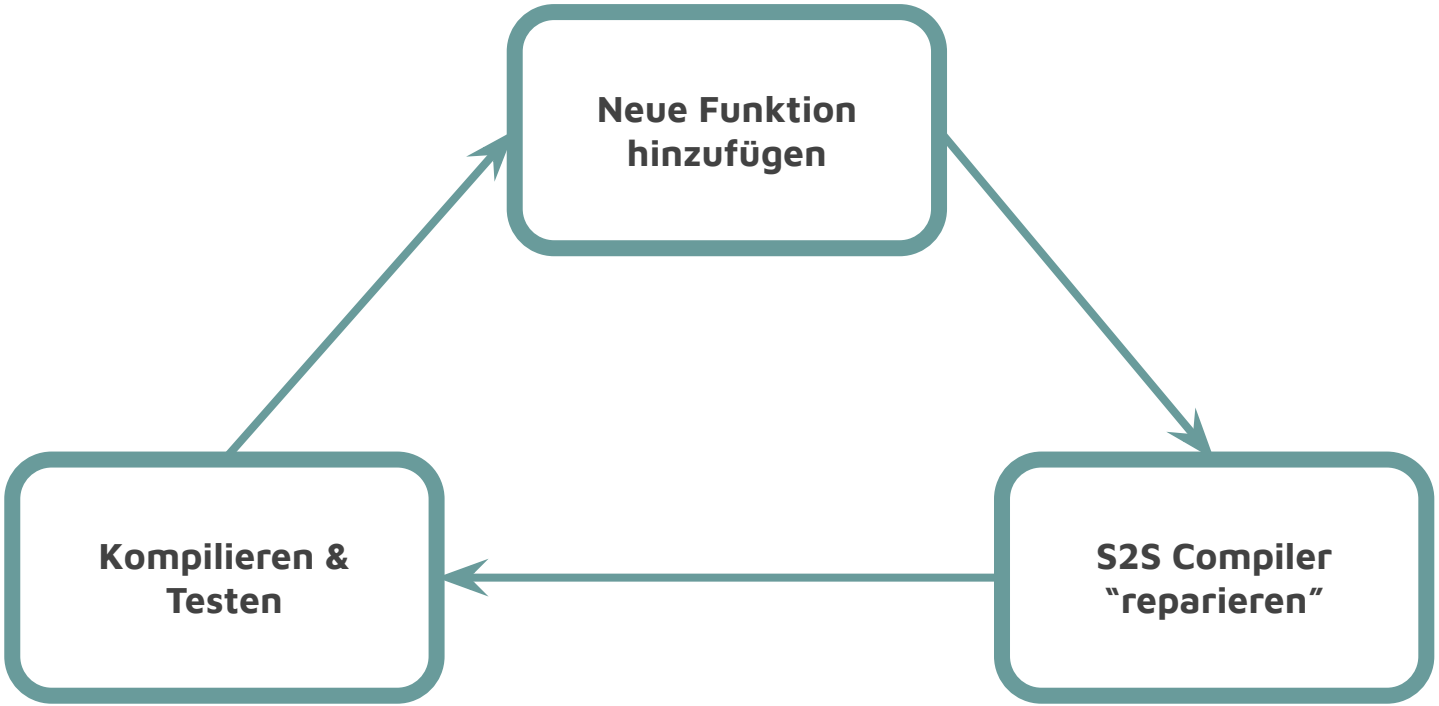
Re: Metamodelle entwickeln



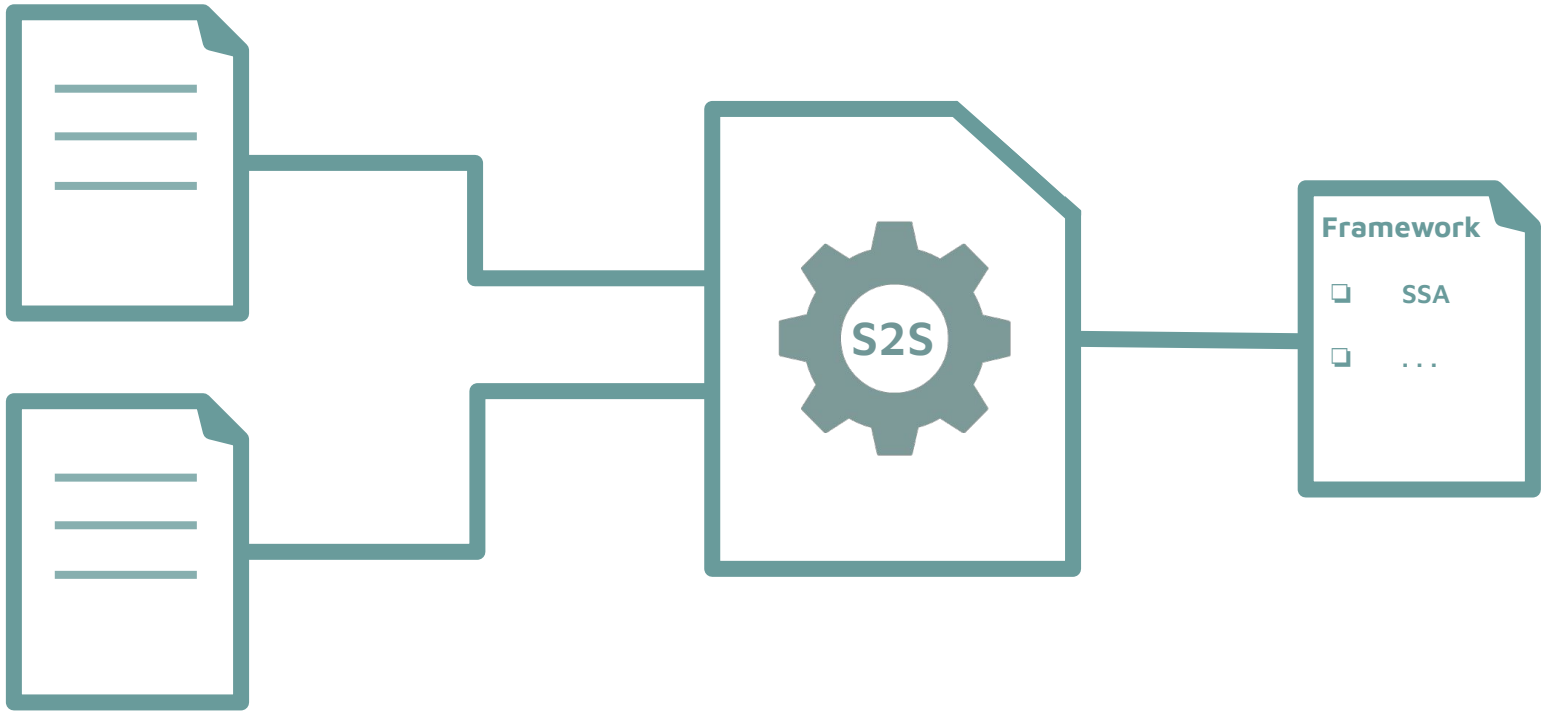
Fragile und Komplexe Arbeitsabläufe



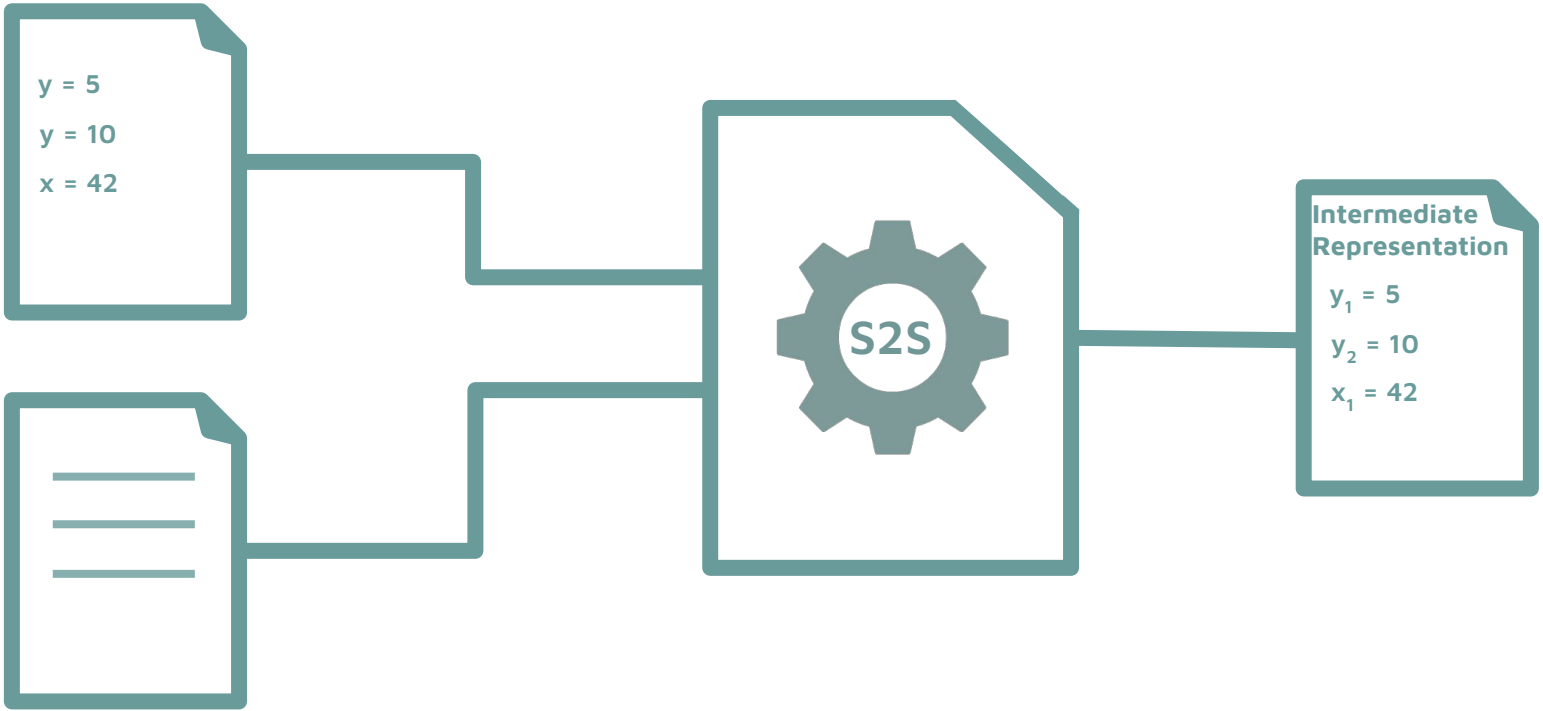
3. Fragile & Komplexe Arbeitsabläufe



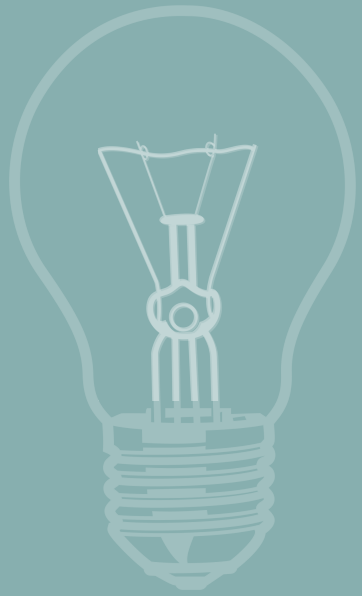
Re: Transformationen normalisieren



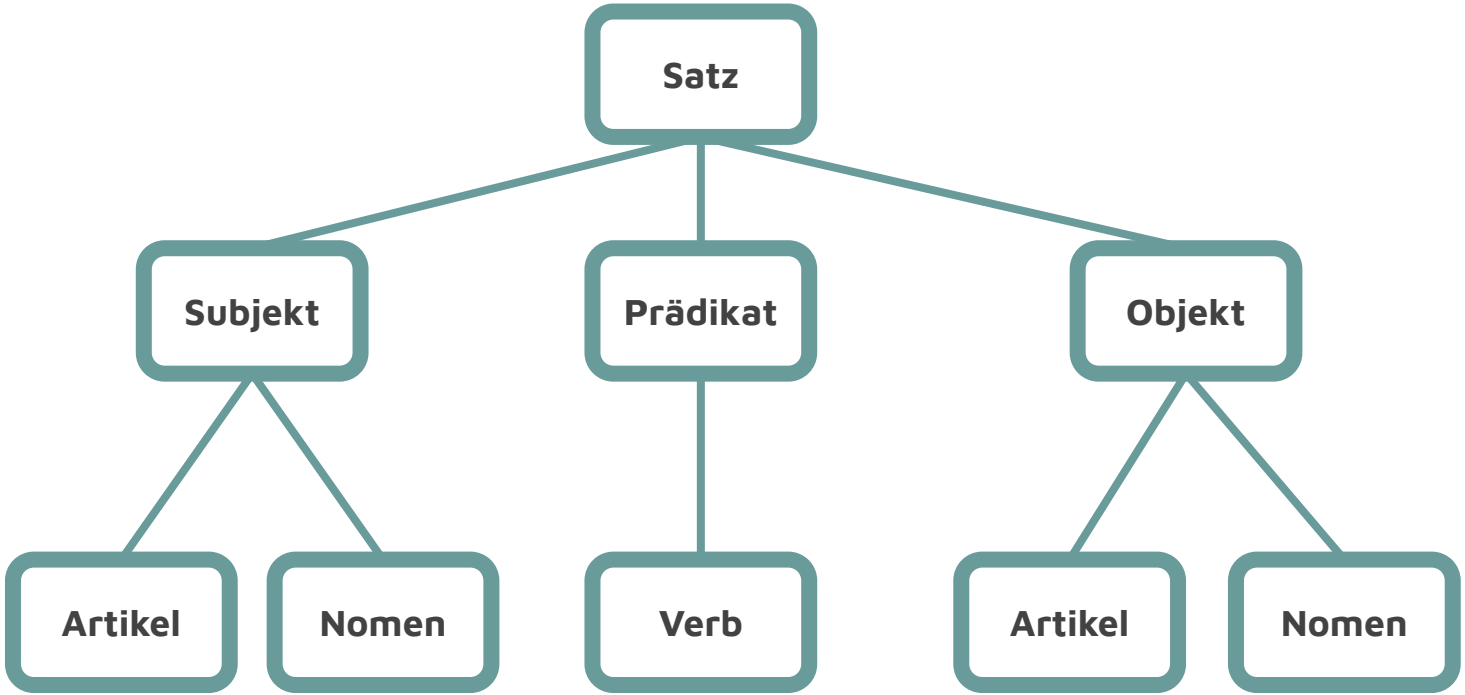
Re: Transformationen normalisieren



Parsing von Natur aus sehr schwer



4. Parsing sehr schwer



4. Parsing sehr schwer

der gefangene floh.

4. Parsing sehr schwer

Der Gefangene floh.

4. Parsing sehr schwer

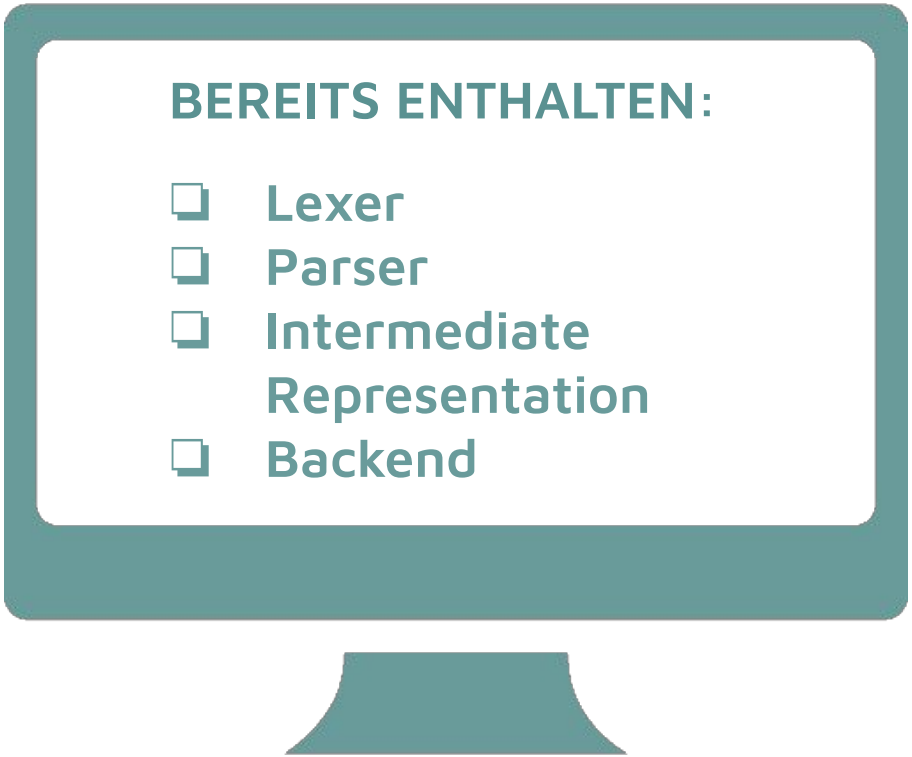
Der gefangene Floh.

4. Parsing sehr schwer

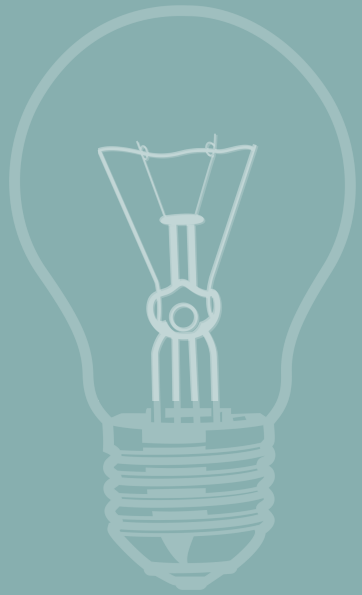
C++

```
int x(y) ;
```

Re: Online Learning Tools entwickeln



Lückenbüßer Lösung

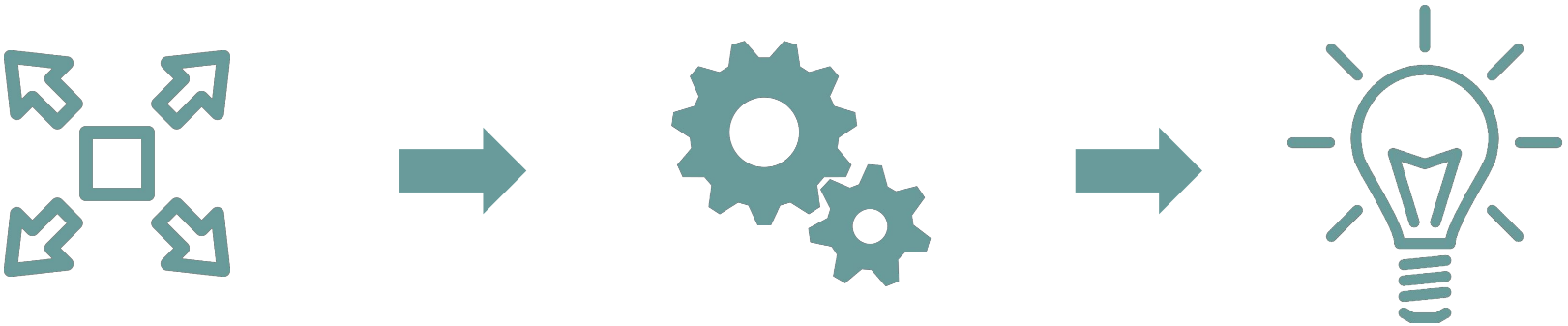


5. Lückenbüßer Lösung

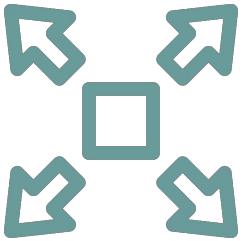
5. Lückenbüßer Lösung

https://www.augsburger-allgemeine.de/panorama/Easyjet-Twitter-Aufruhr-wegen-Foto-Mechaniker-repariert-Flugzeug-mit-Klebeband-id34259672.html

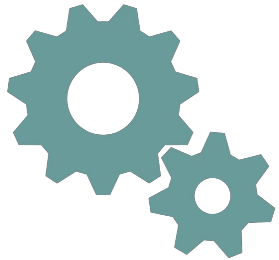
Re: Anwendbarkeit erweitern



Re: Anwendbarkeit erweitern



Anwendbarkeit erweitern



Support wird verbessert

einfacher Lücken*füller*

Verwandte Arbeit

Source-to-Source Parallelization Compilers for Scientific Shared-Memory Multi-core and Accelerated Multiprocessing: Analysis, Pitfalls, Enhancement and Potential

S2S Parallelisierungs Compiler für HLR: Analyse und Bewertung

Verwandte Arbeit

- Harel et al.
- Stärken und Schwächen verschiedener S2S Compiler für automatische Parallelisierung
- 3 S2S Compiler untereinander verglichen

Schlussfolgerung

Validität der Studie

- keine diversifizierende Überprüfung der Gruppierungen
- Viele Gründe wurden nicht beschrieben
- Surveys und Interviews könnten ein klareres Bild ermitteln

Schlussfolgerung

- großer Korpus Paper → S2S & HLR
- Teilmenge extrahiert → Gründe gegen S2S
- Probleme zusammengefasst → Lösungen vorgeschlagen
- Hoffnung → Hilfreich für S2S Community

Eigene Meinung

- Vorteile wurden nicht wissenschaftlich erarbeitet
- Validität durch gesammelte Gründe eingeschränkt
- durch Abstraktion der Probleme, zu abstrakte Lösungen
- gute Zusammenfassung der Vor- und Nachteile von S2S

Negative Eindrücke über die Anwendbarkeit von Source-to-Source Compilern im Hochleistungsrechnen

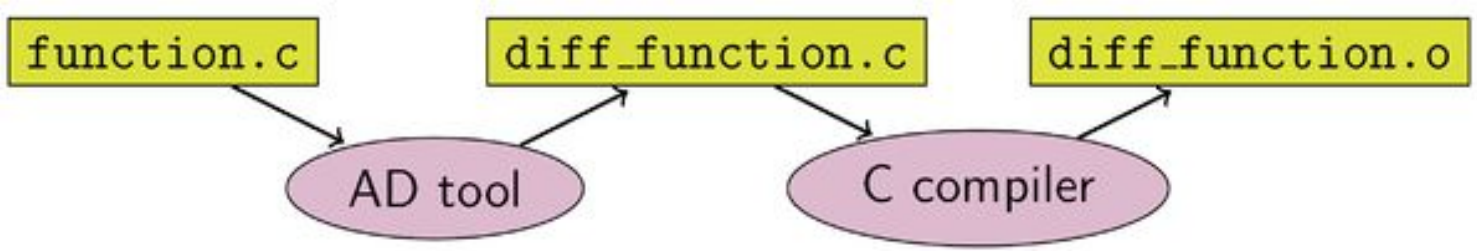
Mariusz Tkocz

Literaturverzeichnis

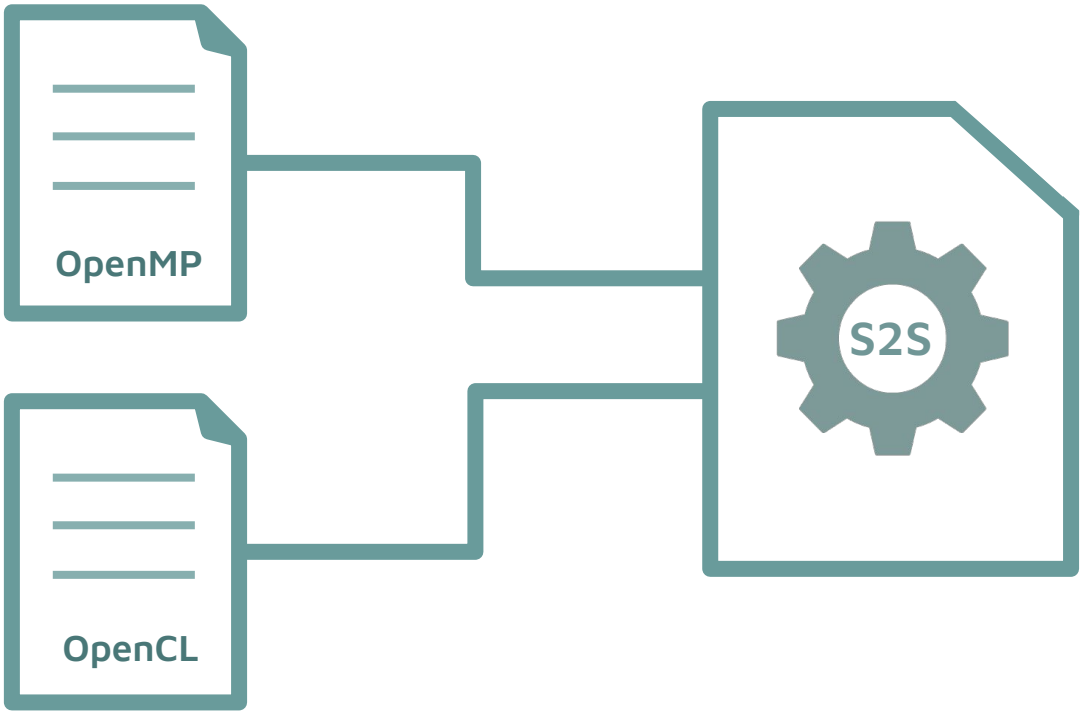
- [1] Milewicz, R., Pirkelbauer, P., Soundararajan, P., Ahmed, H., Skjellum, T. (2021). Negative Perceptions About the Applicability of Source-to-Source Compilers in HPC: A Literature Review. In: Jagode, H., Anzt, H., Ltaief, H., Luszczek, P. (eds) High Performance Computing. ISC High Performance 2021. Lecture Notes in Computer Science(), vol 12761. Springer, Cham. https://doi.org/10.1007/978-3-030-90539-2_16

- [2-54] References in R. Milewicz et al: Negative Perceptions About the Applicability of Source-to-Source Compilers in HPC: A Literature Review. p. 243-246 [1]

Automatisches Differenzieren



Konvertierung Parallelisierungskonstrukte und Bibliotheken



1. Verhindert Compiler Optimierung

- Verhindert problematische Optimierungen und macht sie somit unsichtbar
- Verhindert Compiler oder Laufzeit Optimierungen (s.o.)
- Nicht immer ist der Source Code erhalten
- Moderne Compiler Mid-Ends machen Transformationen rückgängig
- Keine Single Static Assignment Rule, diese hilft um Data Dependencies zu finden und den Program Flow zu untersuchen (Debugging)
- Keine Kontrolle über native Instruktionen oder allokierte Register
- Toolchains die keine Informationen über Komponenten austauschen können

2. Schwer zu Erweitern

- Zugeschnitten für spezifische Schemata (bspw. Sequential C to CUDA)
- Zukünftige Belastung durch Wartbarkeit (neuste Version von CUDA)
- S2S kann nur mit einer Untermenge von CUDA und PTX umgehen
- S2S Tools sind fein abgestimmt, dies limitiert die Generalität der Lösung
- Abhängigkeit von nicht up-to-date S2S Compilern ist ein Entwicklungsrisiko
- Features wie Untied Tasks in OpenMP sind schwer durch S2S zu managen
- Keiner der Tools kann intelligent mit parallelen Programmierungsmodellen umgehen

3. Fragile & Komplexe Arbeitsabläufe

- Zu fragil für die Benutzung in der echten Welt
- Domain Specific Languages benötigen eine lange Entwicklung um die notwendige Reife und Performanz wie andere Compiler zu erreichen
- Kleine Änderungen in Kontrollstrukturen kann vorherige Profiling Runs unbenutzbar machen
- Keine Plattform Unabhängigkeit, Änderungen in der Compile Chain notwendig
- Lösung hat weniger sich bewegende Teile ohne S2S

4. Parsing sehr schwer

- Code durch low-level IRs angehen ist leichter als auf Source Code Ebene
- Inhärente Komplexität von Parsing von Sprachen Konstrukten (C++)
- Problematisch wenn Code komplexe preprocessor macros oder language features hat, die nicht gut vom Parser unterstützt sind
- Nichtverfügbarkeit von Source Code ist eine Hürde

5. Lückenbüßer Lösung

- Meta Programming gibt mehr Flexibilität, indem es Konzepte verständlich darstellt und dem Programmierer die Möglichkeit gibt Optimierungen vorzunehmen, die der Compiler nicht vornehmen konnte
- DSL könnten durch alleinige Benutzung von Macros implementiert werden
- Seit die Unterstützung eingeführt wurde, ist S2S nicht mehr notwendig
- AMPI kann die gleiche Qualität an Ergebnissen erzeugen, wie das Ändern des Source Codes