

BENCHCLOUD, FM-WECK, and FM-TOOLS: Distributed Benchmarking in Containers of Long-Term Conserved Tools

git: <https://gitlab.com/sosy-lab/benchmarking/fm-tools>
web: <https://fm-tools.sosy-lab.org>

Dirk Beyer
LMU Munich, Germany

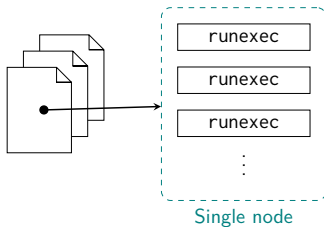
October 29, 2025, at Dagstuhl Seminar [25441](#)

Vision

- ▶ Elastic Cloud for Running Experiments
- ▶ Formal-Methods Tools as Executable Components to be Used in Bigger Systems
- ▶ Library of Information about Tools
- ▶ Containerized Execution

BenchCloud — Motivation

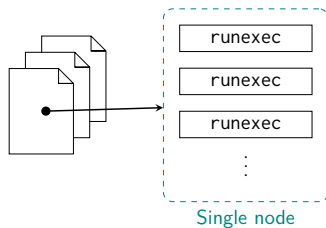
BENCHEXEC [1]



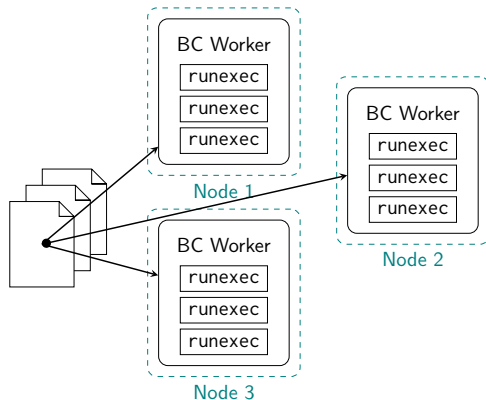
Proc. ASE 2024, with Po-Chun Chien and Marek Jankola

BenchCloud — Motivation

BENCHEXEC [1]



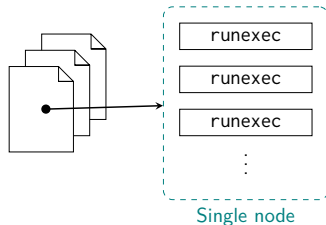
BENCHCLOUD [2]



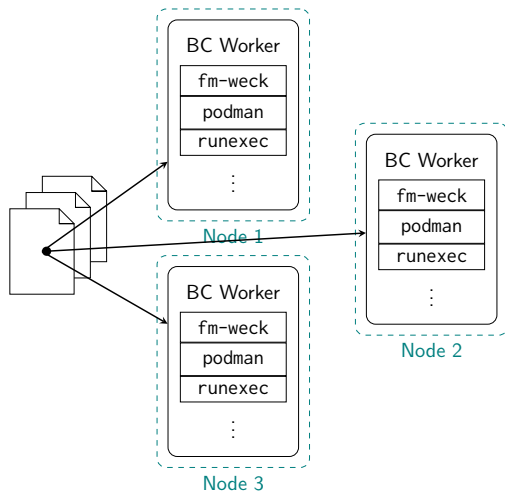
Proc. ASE 2024, with Po-Chun Chien and Marek Jankola

BenchCloud — Motivation

BENCHEXEC [1]

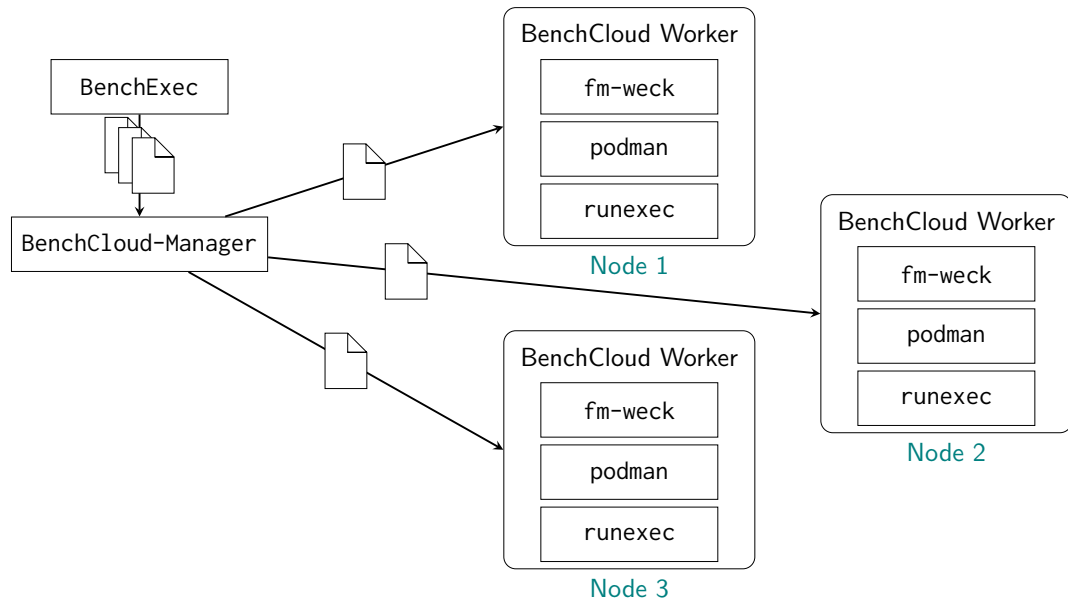


BENCHCLOUD [2] + FM-WECK [3]

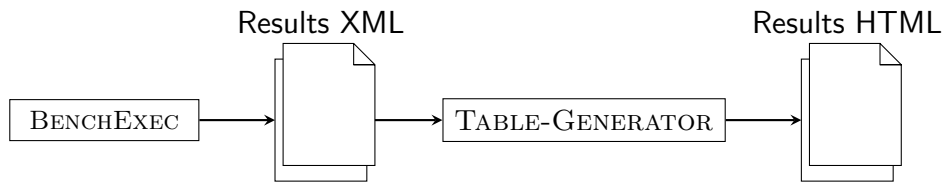


Proc. ASE 2024, with Po-Chun Chien and Marek Jankola

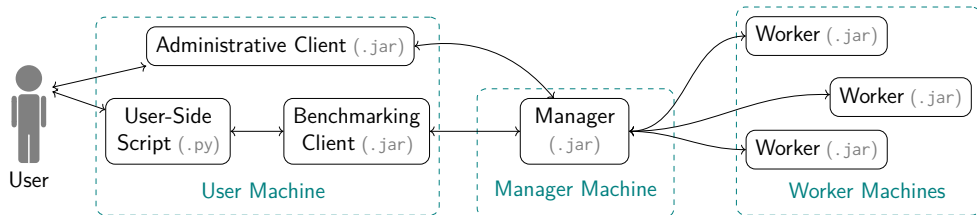
Benchmarking Overview



Benchmarking Overview

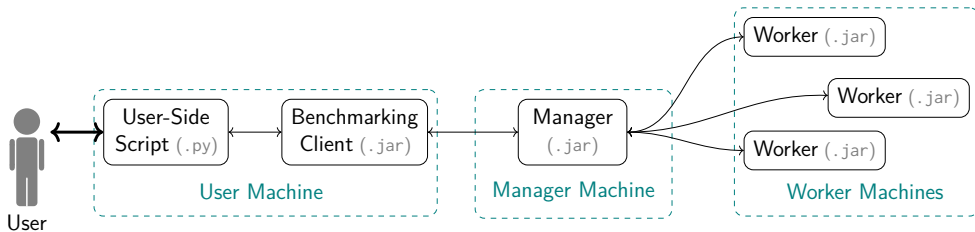


Overview of BenchCloud



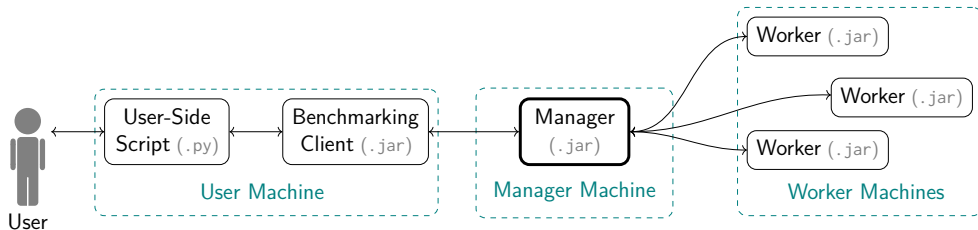
- ▶ Reliable and scalable performance benchmarking
- ▶ Seamless integration with `BENCHEXEC` [1]
- ▶ Used by `SMT-COMP` [4], `SV-COMP` [5], and `Test-Comp` [6]

Benchmarking: Inputs and Outputs



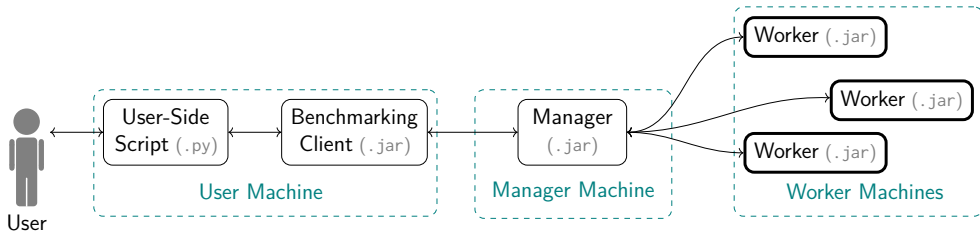
- ▶ Inputs:
 - ▶ Benchmark definitions: resource limits, CPU specs, tasks, tool configs
 - ▶ Precompiled tool executables
- ▶ Outputs: postprocessing and visualization with `BENCHEXEC`'s table-generator [1]

Benchmarking: BenchCloud Manager



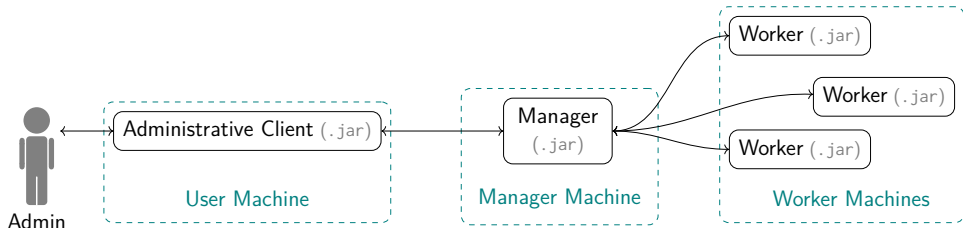
- ▶ Distribute runs to nodes aligning with user's specifications
- ▶ Schedule runs according to priorities
- ▶ Aggregate benchmarking results from the workers

Benchmarking: Workers



- ▶ Start each benchmark run with `runexec`
- ▶ Precise resource management via `cgroups`
- ▶ Isolated and containerized execution via `namespaces`

System Administration



- ▶ Add/remove worker nodes
- ▶ Monitor worker status
- ▶ Renice/cancel benchmark runs

Use Cases

- ▶ International tool competitions in automated reasoning: [SMT-COMP](#) [4], [SV-COMP](#) [5], and [Test-Comp](#) [6]
- ▶ CPAchecker's [7] nightly regression tests



- ▶ 4 running instances at European research institutes

BenchCloud — Summary

- ▶ Facilitate large-scalable benchmarking
- ▶ Easy to use and manage
- ▶ Running system:
<https://benchcloud.sosy-lab.org/>
- ▶ Source code available on Zenodo [8]
- ▶ Supports tool execution in customizable containers using FM-WECK [3]



FM-Tools: Some Steps Towards the Vision regarding Information

- ▶ **Find:** Which tools for software verification exist?
- ▶ ... for test-case generation?
- ▶ ... for SMT solving?
- ▶ ... for hardware verification?
- ▶ **Reuse:** How to get executables?
- ▶ Where to find documentation?
- ▶ Am I allowed to use it?
- ▶ How to use them?
- ▶ **Conserve:** Which operating system, libraries, environment?

Requirements for Solution

- ▶ Support documentation and reuse
- ▶ Easy to query and generate knowledge base
- ▶ Long-term availability/executability of tools
- ▶ Must come with tool support
- ▶ Approach must be compatible with competitions

Solution [9]

One central repository:

<https://gitlab.com/sosy-lab/benchmarking/fm-tools> which gives information about:

- ▶ Location of the tool (via DOI, just like other literature)
- ▶ License
- ▶ Contact (via ORCID)
- ▶ Project web site
- ▶ Options
- ▶ Requirements (certain Docker container / VM)
- ▶ Limits

Maintained by formal-methods community

Example: Entry for CPAchecker [10]

id: cpcachecker

name: CPAchecker

description: |

CPAchecker is a configurable framework for software verification that

is based on configurable program analysis and

implements many model-checking algorithms

to check for software errors and to verify program properties.

input_languages:

- C

project_url: <https://cpcachecker.sosy-lab.org>

repository_url: <https://gitlab.com/sosy-lab/software/cpcachecker>

spdx_license_identifier: Apache-2.0

benchexec_toolinfo_module: benchexec.tools.cpcachecker

fmttools_format_version: "2.0"

fmttools_entry_maintainers:

- dbeyer
 - ricffb
 - PhilippWendler
-

Example: CPACHECKER's Contacts

maintainers:

- **orcid**: 0000-0003-4832-7662
name: Dirk Beyer
institution: LMU Munich
country: Germany
url: <https://www.sosy-lab.org/people/dbeyer/>
 - **orcid**: 0000-0002-5139-341X
name: Philipp Wendler
institution: LMU Munich
country: Germany
url: <https://www.sosy-lab.org/people/wendler/>
-

Example: CPAchecker's Versions

versions:

- `version`: "4.0"
`doi`: 10.5281/zenodo.14203369
`benchexec_toolinfo_options`: ["--svcomp25", "--heap",
"10000M", "--benchmark", "--timelimit", "900_s"]
`required_ubuntu_packages`:
 - `openjdk-17-jdk-headless``base_container_images`:
 - `docker.io/ubuntu:22.04`
- `version`: "4.0-validation-correctness"
`doi`: 10.5281/zenodo.14203369
`benchexec_toolinfo_options`: ["--witness", "\${witness}",
"--correctness-witness-validation", "--heap", "5000m",
"--benchmark", "--option",
"witness.checkProgramHash=false", "--option",
"cpa.predicate.memoryAllocationsAlwaysSucceed=true"]
`required_ubuntu_packages`:
 - `openjdk-17-jdk-headless``base_container_images`:
 - `docker.io/ubuntu:22.04`

Example: CPAchecker's Documentation

literature:

- doi: 10.1007/978-3-031-71177-0_30
title: "Software_Verification_with_CPAChecker_3.0:_Tutorial_and_User_Guide"
year: 2024
 - doi: 10.1007/978-3-642-22110-1_16
title: "CPAChecker:_A_Tool_for_Configurable_Software_Verification"
year: 2011
 - doi: 10.1007/s10817-017-9432-6
title: "A_Unifying_View_on_SMT-Based_Software_Verification"
year: 2018
-

Example: CPAchecker's Web-Page Entry

fm-tools.sosy-lab.org/#tool-cpachecker

Tools for Formal Methods: Tools

Tools Techniques Competitions Frameworks Input Languages Documentation of the YAML Schema ↗

Code on  GitLab

Table of Contents

2LS
aise
AProVE (KoAT + LoAT)
BLAST
BRICK
Bubaak
Bubaak-SpLit
CADP
CBMC
cetfuzz
COASTAL
ConcurrentWitness2Test
CoOpenRace
CoVeriTeam-Verifier-AlgoSelection
CoVeriTeam-Verifier-ParallelPortfolio
CoVeriTest
CPA-BAM-BnB
CPA-BAM-SMG
CPA-witness2test
CPAchecker
CPALockator
CProver-witness2test
CPV
Crux
CSeq
Dartagnan
Deagle
DIVINE
EBF
EmergenTheta
ESBMC-incr
ESBMC-kind
FDSE

CPAchecker

CPAchecker is a configurable framework for software verification that is based on configurable program analysis and implements many model-checking algorithms to check for software errors and to verify program properties.

Project URL: <https://cpachecker.sosy-lab.org>

Repository URL: <https://gitlab.com/sosy-lab/software/cpachecker>

Maintainers: •  Dirk Beyer •  Philipp Wendler

Supported input languages: • C

License: • Apache-2.0

Supported techniques: • Algorithm Selection • ARG-Based Analysis • Automata-Based Analysis • Bit-Precise Analysis • Bounded Model Checking • CEGAR • Concurrency Support • Explicit-Value Analysis • Interpolation • k-Induction • Lazy Abstraction • Numeric Interval Analysis • Portfolio • Predicate Abstraction • Property-Directed Reachability • Ranking Functions • Separation Logic • Shape Analysis • Symbolic Execution

Used frameworks / solvers: • Apron • CPAchecker • JavaSMT • MathSAT

Releases: • 4.0 • 4.0-validation-correctness • 4.0-validation-violation • 2.3.1 • 2.3 • svcomp24-correctness • svcomp24-violation • 2.2 • svcomp22 • 2.1

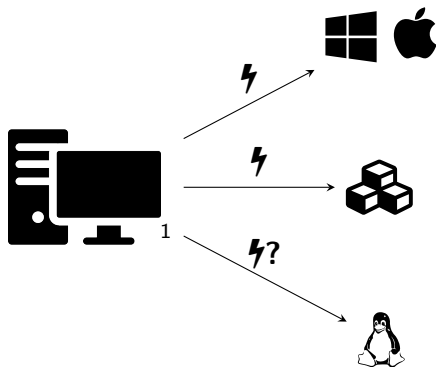
Literature: •  *Software Verification with CPAchecker 3.0: Tutorial and User Guide*. 2024. DOI: 10.1007/978-3-031-71177-0_30
•  *CPAchecker: A Tool for Configurable Software Verification*. 2011. DOI: 10.1007/978-3-642-22110-1_16
•  *A Unifying View on SMT-Based Software Verification*. 2018. DOI: 10.1007/s10817-017-9432-6
•  *CPAchecker 2.3 with Strategy Selection (Competition Contribution)*. 2024. DOI: 10.1007/978-3-031-57256-2_21
•  *CPA-RefSel: CPAchecker with Refinement Selection (Competition Contribution)*. 2016. DOI: 10.1007/978-3-662-49674-9_59
•  *CPAchecker with Support for Recursive Programs and Floating-Point Arithmetic (Competition Contribution)*. 2025. DOI: 10.1007/978-3-031-71177-0_30

October 29, 2025, at Dagstuhl Seminar 25441

FM-Tools is FAIR

- ▶ **F**indable:
overview is available on internet,
generated knowledge base
- ▶ **A**ccessible:
data retrievable via Git, format is YAML
- ▶ **I**nteroperable:
Format is defined in schema,
archives identified by DOIs, researchers by ORCIDs
- ▶ **R**eusable:
Data are CC-BY, each tool comes with a license,
format of tool archive standardized

What about the Environment?

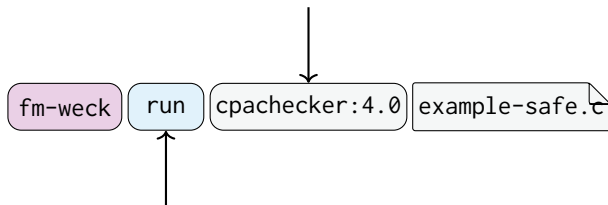


¹Image: Flaticon.com

FM-WECK: Run Tools in Conserved Environment

[3, Proc. FM 2024, with Henrik Wachowitz]

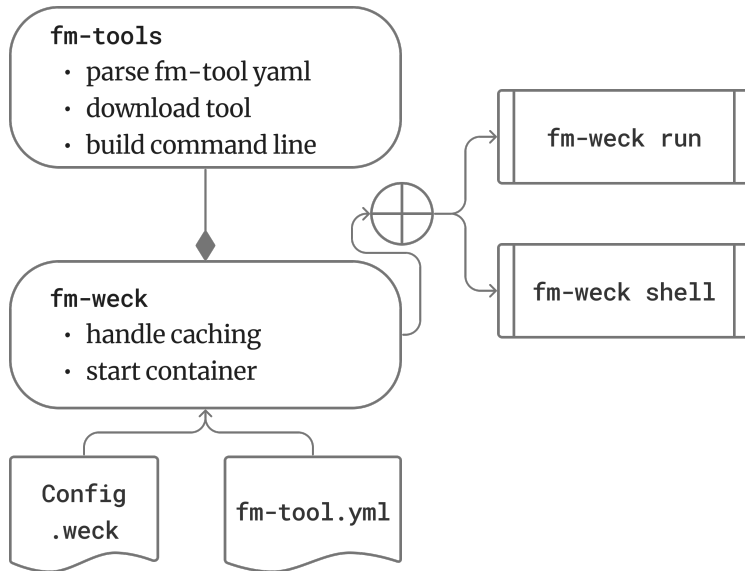
Refer to known fm-tools by
name:version

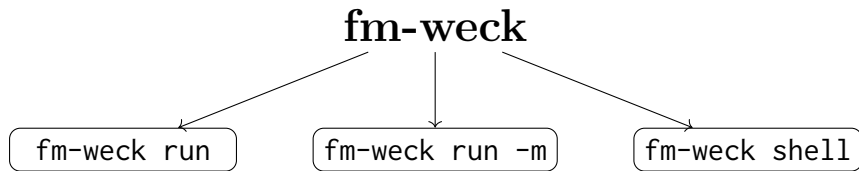


Download, Install and run the
tool

- ▶ No knowledge of the tools CLI needed
- ▶ Tool runs in a container (no dependencies on host system)

FM-WECK: Architecture





- ▶ Download and execute tool in container
- ▶ No knowledge of tool needed
- ▶ Download and execute tool in container
- ▶ Expert knowledge about tool required
- ▶ Spin up interactive shell in tool environment

Conclusion FM-Tools and FM-Weck

FM-TOOLS collects and stores essential information to:

- ▶ Generate a knowledge base about formal-methods tools [9]
<https://fm-tools.sosy-lab.org>
- ▶ Conserve tool versions and their required environment
(with help by Zenodo and Podman/Docker)
- ▶ Run a tool in conserved environment via FM-WECK [3]
- ▶ Please add your tool



<https://fm-tools.sosy-lab.org>

References I

- [1] Beyer, D., Löwe, S., Wendler, P.: Reliable benchmarking: Requirements and solutions. *Int. J. Softw. Tools Technol. Transfer* **21**(1), 1–29 (2019). doi:10.1007/s10009-017-0469-y
- [2] Beyer, D., Chien, P.C., Jankola, M.: BENCHCLOUD: A platform for scalable performance benchmarking. In: *Proc. ASE*. pp. 2386–2389. ACM (2024). doi:10.1145/3691620.3695358
- [3] Beyer, D., Wachowitz, H.: FM-WECK: Containerized execution of formal-methods tools. In: *Proc. FM*. pp. 39–47. LNCS 14934, Springer (2024). doi:10.1007/978-3-031-71177-0_3
- [4] Weber, T., Conchon, S., Déharbe, D., Heizmann, M., Niemetz, A., Reger, G.: The SMT competition 2015-2018. *J. Satisf. Boolean Model. Comput.* **11**(1), 221–259 (2019). doi:10.3233/SAT190123

References II

- [5] Beyer, D.: State of the art in software verification and witness validation: SV-COMP 2024. In: Proc. TACAS (3). pp. 299–329. LNCS 14572, Springer (2024). doi:10.1007/978-3-031-57256-2_15
- [6] Beyer, D.: Software testing: 5th comparative evaluation: Test-Comp 2023. In: Proc. FASE. pp. 309–323. LNCS 13991, Springer (2023). doi:10.1007/978-3-031-30826-0_17
- [7] Beyer, D., Keremoglu, M.E.: CPACHECKER: A tool for configurable software verification. In: Proc. CAV. pp. 184–190. LNCS 6806, Springer (2011). doi:10.1007/978-3-642-22110-1_16
- [8] Beyer, D., Chien, P.C., Jankola, M.: BENCHCLOUD release 1.1. Zenodo (2024). doi:10.5281/zenodo.13742756

References III

- [9] Beyer, D.: Find, use, and conserve tools for formal methods. In: Proc. Festschrift Podelski 65th Birthday. Springer (2024).
https://www.sosy-lab.org/research/pub/2024-Podelski65.Find_Use_and_Conserve_Tools_for_Formal_Methods.pdf
- [10] Baier, D., Beyer, D., Chien, P.C., Jakobs, M.C., Jankola, M., Kettl, M., Lee, N.Z., Lemberger, T., Lingsch-Rosenfeld, M., Wachowitz, H., Wendler, P.: Software verification with CPACHECKER 3.0: Tutorial and user guide. In: Proc. FM. pp. 543–570. LNCS 14934, Springer (2024).
[doi:10.1007/978-3-031-71177-0_30](https://doi.org/10.1007/978-3-031-71177-0_30)