Choiceless Polynomial Time with Witnessed Symmetric Choice

Moritz Lichter
TU Darmstadt, Germany
lichter@mathematik.tu-darmstadt.de

Pascal Schweitzer
TU Darmstadt, Germany
schweitzer@mathematik.tu-darmstadt.de

ABSTRACT
We extend Choiceless Polynomial Time (CPT), the currently only remaining promising candidate in the quest for a logic capturing Ptime, so that this extended logic has the following property: for every class of structures for which isomorphism is definable, the logic automatically captures Ptime.

For the construction of this logic we extend CPT by a witnessed symmetric choice operator. This operator allows for choices from definable orbits. But, to ensure polynomial time evaluation, automorphisms have to be provided to certify that the choice set is indeed an orbit.

We argue that, in this logic, definable isomorphism implies definable canonization. Thereby, we remove the non-trivial step of extending isomorphism definability results to canonization. This step was a part of proofs that show that CPT or other logics capture Ptime on a particular class of structures. The step typically required substantial extra effort.

CCS CONCEPTS
\- Theory of computation \- Finite Model Theory; Complexity theory and logic.

KEYWORDS
Choiceless Polynomial Time, isomorphism, canonization, symmetric choice, logic for PTime

ACM Reference Format:
Moritz Lichter and Pascal Schweitzer. 2022. Choiceless Polynomial Time with Witnessed Symmetric Choice. In 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS ’22), August 2–5, 2022, Haifa, Israel. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3531130.3533348

1 INTRODUCTION
One of the central open problems in descriptive complexity theory is the quest for a logic capturing PTIME [14]. This long-standing open problem [4] asks for a logic in which precisely all polynomial-time decidable properties can be expressed as a sentence and for which all formulas can be evaluated in polynomial time. The alternative would be to prove the nonexistence of such a logic, which would however imply a separation of PTIME and NPTIME [10].

While the general quest for a PTIME-logic remains wide open, progress generally comes in one of two flavors: research results either show that some logic captures PTIME for a more extensive class of structures, or a logic is separated from PTIME, ruling it out as a candidate for a PTIME logic. In this paper we are concerned with a third flavor, namely with reducing the question to a presumably simpler one.

Historically, almost all results showing that a logic captures PTIME for some graph class exploit the Immerman-Vardi theorem [20] which states that inflationary fixed-point logic IFP captures PTIME on ordered graphs. To apply this theorem to a class of unordered graphs, one defines canonization of that class inside a logic (which at least captures IFP). Defining canonization is the task of defining an isomorphic, ordered copy of the input graph. In algorithmic contexts, canonization is closely related to the problem of isomorphism testing. While polynomial time canonization provides polynomial time isomorphism testing, a reduction the other way is unknown. Granted, for most graph classes for which a polynomial time isomorphism algorithm is known, a canonization algorithm is known as well (see [30]). However, we know of no reduction that is universally applicable. Studying the analogous relationship for logics, we are interested in the question of whether definability of the isomorphism problem within a PTIME logic for some graph class provides us with a logic for all of PTIME on that graph class.

Since it is not known that graph isomorphism is solvable in polynomial time in general, it is of course not clear that in a logic for PTIME the isomorphism problem needs to be definable. However, for all graphs on which a logic has been shown to capture PTIME, a polynomial time isomorphism testing algorithm is known [15, 16, 23, 31]. Crucially, in every PTIME-capturing logic for such graph classes isomorphism testing has to be definable. With a reduction from canonization we then obtain a necessary and sufficient condition, namely the definability of the isomorphism problem. Regarding the issue of isomorphism versus canonization, we highlight that defining canonization often requires considerably more effort than defining the isomorphism problem [15, 16].

In this paper we present a logic in which a definable isomorphism test automatically implies a definable canonization. After rank logic was recently eliminated as a candidate of a logic capturing PTIME [22], we consider an extension of the one major remaining candidate, namely Choiceless Polynomial Time (CPT) [1]. The logic CPT operates on hereditarily finite sets formed from the vertices of the input graph. The construction of these sets is isomorphism-invariant, which guarantees that every CPT term or formula evaluates to an isomorphism-invariant result. This is generally regarded as a requirement for a reasonable logic [18]. The requirement has an important consequence: while in algorithms it is common to make choices that are not necessarily isomorphism...
invariant (e.g., pick the first neighbor of a vertex within a DFS-transversal and then process it), this cannot be done in CPT – one has to process all possible choices in equal fashion. For algorithms making choices, we have to prove that they compute the correct (in particular isomorphism-invariant) result, but in a logic this property should be built-in. One possibility to overcome this problem was studied by Gire and Hoang [11] as well as by Dawar and Richerby [7]. They extended IFP with a symmetric choice construct, which allows that during a fixed-point computation in every step one element can be chosen from a set that has been defined. But, in order to ensure isomorphism-invariance, these choice sets have to be orbits of the graph. The output of such a fixed-point computation with choices is not necessarily isomorphism-invariant. However, at least we are guaranteed that all possible outputs are related via automorphisms of the graph, independent of the choices that were made. Crucially, the logics are designed so that fixed-point computations are used only as “intermediate results” to overall finally define a property. Since this property is either true or false, the output of a formula is isomorphism-invariant.

While this approach of introducing symmetric choices yields a reasonable logic, it is not clear whether its formulas can be evaluated in polynomial time. Indeed, when a choice is to be made, it has to be verified that the choice set is actually an orbit and it is not known that orbits can be computed in polynomial time. This is resolved in [11] by handing over the obligation to check that the choice sets are orbits to the formulas themselves. For this, the formulas, beside defining the choice set, also have to define automorphisms which can be used to check whether the choice set is indeed an orbit. That way, the logic can be evaluated in polynomial time. We apply a similar approach to CPT. A fixed-point operator is added, in which in every iteration a choice is made from a choice set. For each choice set, automorphisms which certify that the choice set is indeed an orbit have to be provided. We call this witnessed symmetric choice (WSC).

So why should witnessed symmetric choice in CPT suffice to show that isomorphism testing and canonization are equivalent? Here we build on two existing results. The first one [17] shows that in CPT a definable isomorphism test implies a definable complete invariant, that is, an ordered object can be defined which is equal for two input structures if and only if they are isomorphic. The second, more classical result is due to Gurevich [19]. It shows how an algorithm computing complete invariants can be turned into an algorithm computing a canonization. This algorithm requires that the class of graphs is closed under individualization (that is, under coloring individual vertices). While being closed under individualization is a restriction in some contexts [21, Theorem 33], this is usually not the case [21, 25]. The canonization algorithm repeatedly uses the complete invariant to compute a canonical orbit, chooses and individualizes one vertex in that orbit, and proceeds until all vertices are individualized. Thereby, a total order on the vertices is defined. And indeed, this algorithm can be expressed in CPT extended by witnessed symmetric choice and a definable complete invariant can be turned into a definable canonization.

**Results.** We extend CPT with a fixed-point operator with witnessed symmetric choice and obtain the logic CPT+WSC. Here some small, but important formal changes to [7, 11] have to be made so that we can successfully implement a variant of Gurevich’s canonization algorithm in CPT+WSC. However, we prefer to have a logic so that a definable isomorphism problem implies the same logic captures \( \text{Ptime} \) rather than in an extension. Showing precisely this property for CPT+WSC turns out to be rather difficult and formally intricate in several aspects. Indeed, we lift a result of [17] from CPT to CPT+WSC thereby showing the following: if CPT+WSC defines isomorphism of a class of structures closed under individualization, then it defines a complete invariant and using the mentioned canonization algorithm CPT+WSC defines a canonization, too. We obtain:

**Theorem 1.1.** If CPT+WSC defines isomorphism of a class of \( \tau \)-structures \( \mathcal{K} \) (which is closed under individualization), then CPT+WSC defines a canonization of \( \mathcal{K} \)-structures and captures \( \text{Ptime} \) on \( \mathcal{K} \)-structures.

Finally, we apply these results to the Cai–Fürer–Immerman query [3] and construct a class of base graphs, for which the CFI-query was not known to be definable in CPT.

**Our Technique.** We ensure that fixed-point operators with witnessed symmetric choice always yield isomorphism-invariant results by following the approach of [11]. Every fixed-point operator comes with a formula, called the output formula, which is evaluated on the (not necessarily isomorphism-invariantly) defined fixed-point. We use the techniques of [7] to define the semantics of this fixed-point operator. The most important difference is that the term producing witnessing automorphism also has access to the defined fixed-point. This turned out to be crucial to implement Gurevich’s canonization algorithm and in turn necessitates other minor differences to [7, 11]. Equipped with these changes, extending Gurevich’s canonization algorithm to provide witnessing automorphism is rather straightforward.

This shows that a definable isomorphism problem in CPT implies that CPT+WSC is a logic capturing \( \text{Ptime} \). For our theorem however, we require that the statement is true whenever isomorphism is definable in CPT+WSC rather than CPT. Therefore, we require that a CPT+WSC-definable isomorphism test implies the existence of a CPT+WSC-definable complete invariant. We therefore extend the DeepWL computation model used in [17] to show the very same statement for CPT to deal with witnessed symmetric choices. The proof of [17] is based on a translation of CPT to DeepWL, a normalization procedure in DeepWL yielding the complete invariant, and a translation back into CPT. Unfortunately, it turned out that this normalization procedure cannot be easily adapted to DeepWL with witnessed symmetric choice. At multiple points we have to change small but essential parts of definitions and so cannot reuse as many results of [17] as one would have liked. The philosophical reason for this is that DeepWL is based on constructing everything in parallel, which is incompatible with choices. We cannot compute with different possible choices at the same time in the same graph, as these choices influence each other. Hence, we nest DeepWL-algorithms to resemble nested fixed-point operators with witnessed symmetric choice.

Omitted proofs and details are given in the full version [24].

**Related Work.** In the quest for a \( \text{Ptime} \) logic, IFP with counting was shown not to capture \( \text{Ptime} \) [3], but used to capture \( \text{Ptime} \).
on various graph classes. These include graphs with excluded minors [15] and graphs with bounded rank width [16]. Both results take the route via canonization. On the negative side, rank logic [22] and the more general linear algebraic logic [6] were separated from Ptime.

CPT was shown to capture Ptime on various classes of structures, for example on paged structures [2] (i.e. on disjoint unions of arbitrary structures and sufficiently large cliques), structures with bounded color class size whose automorphism groups are abelian [31], and on (some) structures with bounded color class size whose automorphism groups are dihedral groups [23]. Also here each of the results is obtained via canonization. Philosophically, all these approaches are somewhat orthogonal to witnessed symmetric choice. They use the fact that some set of objects, for which it is not known whether they form orbits, is small enough to try out all possible choices. The CFI-query on ordered base graphs was shown to be CPT-definable [9] using deeply nested isomorphism-invariant sets. This was generalized to graphs with logarithmic color class size and to graphs with linear maximal degree [28]. In CPT+WSC defining the CFI-query on ordered base graphs is comparatively easier similar to IFP with witnessed symmetric choice in [11]. While it is still open whether CPT captures Ptime, there are isomorphism-invariant functions not definable in CPT [29], see also [26] for recent work on limits of definability in CPT.

The extension of first order logic with non-witnessed symmetric choice was studied in [7, 11]. Whenever the choice set is not an orbit, nothing is chosen. The more general variant in [7] supports parameters for fixed-point operators with symmetric choice and as such allows fixed-point operators to be nested. We followed most of these approaches and generalized the usage of quantifiers to output formulas in order to wrap the calculated fixed-point in an isomorphism-invariant output. While in the first-order setting the fixed-point operators are tied to define relations, in the CPT setting we can of course define arbitrary hereditarily finite sets. For these sets output formulas seem more suitable than solely quantifiers. Dropping the requirement of choosing from orbits is also studied in [8], called nondeterministic choice. When only formulas are considered which always produce a deterministic result, one captures Ptime. However, this does not yield a reasonable logic because this property of formulas is undecidable.

2 PRELIMINARIES

We denote with [k] the set \{1, \ldots, k\} and the i-th entry of a tuple i ∈ N^k (for some set N) with i_i.

A (relational) signature τ = \{E_1, \ldots, E_ℓ\} consists of a set of relation symbols with associated arities n_i ∈ N for all i ∈ [ℓ]. A τ-structure A is a tuple A = (A, E_1^A, \ldots, E_ℓ^A) where E_i^A ⊆ A^{n_i} for all i ∈ [ℓ]. We denote the universe of A by A, call its elements atoms, and only consider finite structures. The disjoint union of two structures A and B is A ∪ B.

The hereditarily finite sets over A, denoted by HF(A), for some set of atoms A is the inclusion-wise smallest set such that A ⊆ HF(A) and a ∈ HF(A) for every finite a ⊆ HF(A). A set a ⊆ HF(A) is transitive, if a ∈ b for some b implies c ∈ a. The transitive closure TC(a) of a is the least (with respect to set inclusion) transitive set b with a ⊆ b.

Let Φ be a τ-structure and ab a tuple of HF(A)-sets. We write Aut(Φ) for the group of automorphisms of Φ and Aut((Φ, a)) for the group of automorphisms Φ which stabilize a. A set b ∈ HF(A) is an orbit of (Φ, a) if b = \{φ(c) | φ ∈ Aut((Φ, a))\} for one (and thus every) c ∈ b. If b is a set of k-tuples of atoms, we call b a k-orbit.

We write orb_b(Φ) for the set of k-orbits of Φ.

Choiceless Polynomial Time

The logic CPT was introduced by Blass, Gurevich, and Shelah [1] using a pseudocode-like syntax. Later there were “logical” definitions using iteration terms or fixed-points. To give a concise definition of CPT, we follow [12] and use ideas of e.g. [27] to enforce polynomial bounds.

Extend a signature τ by adding set-theoretic function symbols \( r^{HF} := τ \cup \{∅, \text{Atoms}, \text{Pair}, \text{Union}, \text{Unique}, \text{Card}\} \), where ∅ and Atoms are constants, Union, Unique, and Card are unary, and Pair is binary. The hereditarily finite expansion HF(Φ) of a τ-structure Φ is a \( r^{HF} \)-structure over the universe HF(A) defined as follows: all relations in τ are interpreted as they are in Φ. The special function symbols have the expected set theoretic interpretation:

- \( \emptyset^{HF}(\cdot) = \emptyset \) and Atoms^{HF}(\cdot) = A,
- Pair^{HF}(a, b) = \{a, b\},
- Union^{HF}(\cdot)(a) = \{b | \exists c ∈ a. b ∈ c\},
- Unique^{HF}(\cdot)(a) = \{b | \text{if } a = \{b\}\}, and
- Card^{HF}(\cdot)(a) = \{a | \text{if } a \notin A\} \cup \{∅ | \text{otherwise}\},

where |a| is encoded as a von Neumann ordinal.

The Unique function is isomorphism-invariant because it only evaluates non-trivially when applied to singleton sets.

The logic CPT is obtained as the polynomial time fragment of the logic BGS (after Blass, Gurevich, and Shelah [1]): A BGS term is composed of variables and function symbols from \( r^{HF} \) and the two following constructs: if s(x, y) and t(x) are terms with a sequence of free variables \( x \) (and an additional free variable y in the case of s) and \( Φ(x, y) \) is a formula with free variables \( x \) and y then \( r(x) = \{s(x, y) | y ∈ t(x), Φ(x, y)\} \) is a comprehension term with free variables \( x \). For a term s(x, y) with free variables \( x \) and y the iteration term \( s(y)^{Φ}(x) \) has free variables \( x \). BGS formulas are obtained as \( E(t_1, \ldots, t_k) \) (for \( E \) ∈ τ of arity k and BGS terms \( t_1, \ldots, t_k \)) as \( t_1 \) = \( t_2 \), and the Boolean connectives.

Let Φ be a τ-structure. BGS terms and formulas are evaluated over HF(A) by the notation \([t]^{HF} : HF(A)^k → HF(A)\) that maps values \( \hat{a} = (a_1, \ldots, a_k) \in HF(A)^k \) for the free variables \( \hat{x} = (x_1, \ldots, x_k) \) of a term t to the value of t obtained if we replace \( x_i \) with \( a_i \) (for all \( i ∈ [k] \)). For a formula Φ with free variables \( \hat{x} \), the notation \([Φ]^{HF} \) is the set of all \( \hat{a} = (a_1, \ldots, a_k) ∈ HF(A)^k \) satisfying Φ.

The denotation of a comprehension term r as above is the following: \([r]^{HF}(\hat{a}) = ([s]^{HF}(\hat{a}) \mid b ∈ [t]^{HF}(\hat{a}), (\hat{a}b) ∈ [Φ]^{HF})\), where \( \hat{a}b \)

\footnote{Here we differ from the definition in [12], in which s is only allowed to have one free variable y. For CPT, allowing more free variables does not increase expressiveness, but for our extensions later it is useful to allow additional free variables in an iteration term.}
denotes the tuple \((a_1, \ldots, a_k, b)\). An iteration term \(s[y]\times(x)\) for a tuple \(b \in HF(A)\) with sets for the free variables defines a sequence of sets via \(a_i := 0\) and \(a_{i+1} := [s[y]\times(b_{a_i})]\). Let \(t := t(s[y]\times, \bar{A}, \bar{b})\) be the least number \(i\) such that \(a_{i+1} = a_i\). If such an \(i\) exists we set \([s[y]\times]\times(b) := a_i\) and we set \([s[y]\times]\times(b) := 0\) otherwise.

A CPT term (or formula, respectively) is a tuple \((t, p)\) (or \((\phi, p)\), respectively) of a BGS term (or formula) and a polynomial \(p(n)\). The semantics of CPT is derived from BGS by replacing \(t\) with \(t\times\phi\) everywhere (or \(t\times\phi\) and \(\Phi\), respectively) such that \(\Phi\) defines the set of universal or isolated vertices in \(G\). Thus, we use a WSC-fixed-point operator to choose one such vertex, remove it, and reduce it to the empty graph by repeatedly removing a universal or isolated vertex. A graph \(G\) is satisfied if the output formula \(\Phi\) is satisfied where \(\Phi\) witnesses a proper subset of an orbit. If for every \(b, c \in N\) there is a \(\varphi \in M\) satisfying \(\varphi(b) = c\).

Note that this definition also allows witnessing proper subsets of orbits. However, the sets \(N\) of interest in the following will always be given by an isomorphism-invariant function and so \(N\) can never be a proper subset of an orbit.

Now fix an arbitrary \(\tau\)-structure \(\mathfrak{A}\) and a tuple \(\bar{a} \in HF(A)^\tau\). Let \(f^\mathfrak{A}_\bar{a}\), \(f^\mathfrak{A}_\bar{a}\times\cdot\), \(HF(A)^\tau \rightarrow HF(A)\) and let \(f^\mathfrak{A}_\bar{a}\times\cdot\) : \(HF(A) \rightarrow HF(A)\) be \(\mathfrak{A}\)-isomorphism-invariant functions.

Define \(T(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\) to be the (possibly infinite) unique least tree whose vertices are labeled with \(HF(A)\)-sets (nodes in the tree may have the same label) for which:

- The root is labeled with \(\emptyset\).
- A vertex labeled with \(b \in HF(A)\) has for every \(c \in f^\mathfrak{A}_\bar{a}(b)\) a child labeled with \(f^\mathfrak{A}_\bar{a}(b, c)\).

Let \(P(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\) be the set of tuples \(p = (b_1, \ldots, b_n)\) of \(HF(A)\)-sets such that \(n \geq 2\), \(b_1 = 0\), \(b_{n-1} = b_n\), \(b_{i+1} \neq b_i\) for all \(1 \leq i < n\), and there is a path of length \(n\) in \(T(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\) starting at the root and the \(i\)-th vertex in the path is labeled with \(b_i\) for all \(i \in [n]\). In other words, \(T(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\) models the computation for all the possible choices and \(P(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\) is the set of all possible labels yielding a fixed-point. We call the elements of \(P(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\) also paths.

The function \(f^\mathfrak{A}_\bar{a}\times\cdot\) witnesses a path \(p = (b_1, \ldots, b_n) \in P(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\) whenever \(f^\mathfrak{A}_\bar{a}(b_i, b_{i})\) witnesses \(f^\mathfrak{A}_\bar{a}(b_i, b_{i+1})\) as an \((\mathfrak{A}, \bar{a}, \ldots, b_i, b_{i+1})\)-orbit for every \(i \in [n-1]\). Finally, we define \(WSC^\mathfrak{A}(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot) := \{b_n | (b_1, \ldots, b_n) \in P(f^\mathfrak{A}_\bar{a}, f^\mathfrak{A}_\bar{a}\times\cdot)\} \cap HF(A)\) and the witnessing term \(\Phi_{out}\) should have a second free variable, that we do not use here.
we thus have to require that choice sets are witnessed, we abort evaluation and output an error, indicating there is even more complicated without providing further insights.

Corollary 3.4. The function \( f_{\text{wit}} \) either witnesses all paths in \( \mathcal{P}(\cdot, \cdot, \cdot, \cdot, \cdot) \) or none of them.

Corollary 3.5. \( \text{WSC}^*(\cdot, \cdot, \cdot, \cdot, \cdot) \) is an \((\mathfrak{F}, \cdot)\)-orbit.

Now we can define the denotation of WSC-fixed-point operators\footnote{In \cite{5} the fixed-point operator with symmetric choice is not evaluated on \( \mathfrak{F} \), but on the reducible \( \mathfrak{P} \).} For a BGS+WSC term \( s \) with free variables \( x_1, \ldots, x_k \) and a tuple \( \bar{a} \in \text{HF}(A)^k \) for \( 1 \leq k \), we write \( \bar{x} \bar{a} \bar{a} \) for the "partial" application of the function \( \bar{x} \bar{a} \bar{a} \) so for the function \( \text{HF}(A)^k \to \text{HF}(A) \) defined by \( \bar{b} \mapsto \bar{a} \bar{b} \).

Let \( s_{\text{step}} \) and \( s_{\text{wit}} \) be BGS+WSC terms with free variables \( \bar{x}_{xy} \), let \( s_{\text{choice}} \) be a BGS+WSC term with free variables \( \bar{x}_{yz} \), and let \( \Phi_{\text{out}} \) be a BGS+WSC formula with free variables \( \bar{z}_{zx} \).

\[
\text{WSC}^* \bar{z}_{zx} \text{.} (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, \Phi_{\text{out}})_{\mathfrak{F}} :=
\begin{cases}
\{ \bar{a} \mid \bar{a} \in \Phi_{\text{out}}_{\mathfrak{F}} \} & \text{for every } \\
\text{for every } \bar{b} \in \text{WSC}^*(s_{\text{step}})_{\mathfrak{F}}(\bar{a}), s_{\text{choice}}_{\mathfrak{F}}(\bar{a}), s_{\text{wit}}_{\mathfrak{F}}(\bar{a})
\end{cases}
\]

Failure on Non-Witnessed Choices. While the denotation defined as above results in a reasonable logic, we want a special treatment of the case when choices cannot be witnessed. Whenever during the evaluation of a formula there is a path in \( \mathcal{P}(\cdot, \cdot, \cdot, \cdot, \cdot) \) that is not witnessed, we abort evaluation and output an error, indicating there was a non-witnessed choice. Formally, we extend the denotation by an error-marker \( \dagger \). Then the denotation of a term becomes a function \( \text{HF}(A)^k \to \text{HF}(A) \cup \{ \dagger \} \) and the denotation of a formula a function \( \text{HF}(A)^k \to \{ \top, \bot, \dagger \} \). Whenever a \( \dagger \) occurs, it is just propagated. We omit formal definitions here.

Fixing Intermediate Steps. We define the evaluation of choice terms similar to \cite{5} using the tree \( T(f_{\text{wit}}, \cdot, \cdot, \cdot, \cdot) \). However, our definition is different in one crucial aspect: In Section 4 it gets crucial that \( f_{\text{wit}} \), in the setting of Lemma 3.3, always gets the fixed-point bound \( b_{\text{hit}} \) as input to witness orbits. In order to prove Lemma 3.3 we thus have to require that choice sets are \( (\mathfrak{F}, \cdot, b_1, \ldots, b_k) \)-orbits, i.e. in BGS+WSC all earlier steps in the fixed-point computation need to be fixed while in \cite{5} only the single previous step \( b_1 \) needs to be fixed.

3.2 CPT+WSC

Similarly to how CPT arises from BGS, we obtain CPT+WSC by enforcing polynomial bounds on BGS+WSC formulas: A CPT+WSC term (formula) is a pair \( (s, \Phi(p(n))) \) of a BGS+WSC term (formula) and a polynomial. For BGS operators, we add the same restrictions as in CPT. For a WSC-fixed-point operator

\[
\text{WSC}^* \bar{z}_{zx} \text{.} (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, \Phi_{\text{out}}), \text{a structure } \mathfrak{F}, \text{and a tuple } \bar{a} \in \text{HF}(A)^k, \text{we restrict the set } \mathcal{P}(\cdot, \cdot, \cdot, \cdot, \cdot) \text{ to paths } (b_1, \ldots, b_k) \text{ of length } k \leq p(|A|) \text{ for which } |T(b_i)| \leq p(|A|) \text{ for all } i \in [k]. \text{If there is a } \bar{b} \text{ in } \mathcal{P}(\cdot, \cdot, \cdot, \cdot, \cdot) \text{ of length greater than } \leq p(|A|) \text{ or in some path there is a set not bounded by } p, \text{the WSC-fixed-point operator has denotation } \dagger.
\]

It is important to note here, that we do not require that \( |\text{WSC}^*(\cdot, \cdot, \cdot, \cdot, \cdot)(\bar{a}), \text{WSC}^*(\cdot, \cdot, \cdot, \cdot, \cdot)(\bar{a}), |\text{WSC}^*(\cdot, \cdot, \cdot, \cdot, \cdot)(\bar{a})| \) is bounded by \( p(|A|) \). In fact using WSC-fixed-point operators only makes sense if the set may be superpolynomially sized, as otherwise we could define it with a regular iteration term. It is also important to output \( \dagger \) and not \( \emptyset \) when the polynomial bound is exceeded because in that case we cannot validate whether all choice sets are orbits (and so it might depend on the choices whether the bound is exceeded or not). To evaluate the witnessing term we need access to the fixed-point, which cannot be computed if the polynomial bound gets exceeded.

Because the WSC-fixed-point operator can only choose from orbits, CPT+WSC is isomorphism invariant.

Lemma 3.6. For every structure \( \mathfrak{F} \), every CPT+WSC term \( s \) and formula \( \Phi \) the denotations \( [s]_{\mathfrak{F}} \) and \( [\Phi]_{\mathfrak{F}} \) are unions of \( \mathfrak{F} \)-orbits.

With Lemma 3.6 we can show that CPT+WSC can be evaluated in polynomial time.

Lemma 3.7. For every CPT+WSC term \((s, \Phi(p(n))) \) or formula \((\Phi, p(n))) \) we can compute in polynomial time on input \( \mathfrak{F} \) and \( \bar{a} \in \text{HF}(A)^k \) the denotation \( [s]_{\mathfrak{F}}(\bar{a}) \) or \( [\Phi]_{\mathfrak{F}}(\bar{a}) \) respectively.

Proof Sketch. The only interesting cases to consider are WSC-fixed-point operators \( \text{WSC}^* \bar{z}_{zx} \text{.} (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, \Phi_{\text{out}}) \). By Corollaries 3.4 and 3.5 we can get at every step of the iteration choose an arbitrary element of the defined choice set and continue. If a fixed-point \( a \in \text{HF}(A) \) is reached within the polynomial time bound (otherwise output \( \dagger \)), check the witnessing term \( s_{\text{wit}} \) whether it witnesses all choices. If this is the case, evaluate \( \Phi_{\text{out}} \) on \( a \), which is justified by Lemma 3.6, and otherwise output \( \dagger \).

The WSC-fixed-point operator can only output truth values. These are, by design, isomorphism-invariant. One can actually use the WSC-fixed-point operator to define \( \text{HF}(\cdot) \)-sets, which are isomorphism-invariant, too. This can be achieved by encoding the \( \text{HF}(\cdot) \)-set as a directed acyclic graph (DAG) with numbers as vertex- and using the WSC-fixed-point operator to define which vertices are adjacent. Then the \( \text{HF}(\cdot) \)-set can be reconstructed from the DAG.

Note that CPT+WSC has two fixed-point operators: iterations terms and WSC-fixed-point operators. Actually both are needed, because the WSC-fixed-point operator cannot define \( \text{HF}(A) \) sets, which ensures isomorphism-invariance.

4 CANONIZATION IN CPT+WSC

From now, we work with classes of relational \( \tau \)-structures \( \mathcal{K} \). We assume that these classes are closed under isomorphisms and only contain connected structures. Since we are interested in isomorphism testing and canonization, the case of unconnected structures reduces to connected ones.

The process of individualizing certain atoms in structures plays a crucial role and we only want to consider classes of structures
closed under individualizing atoms. The formulas we are going to define in fact iteratively individualize atoms. So it will be convenient to capture the individualized atoms by CPT+WSC “internal” tuples, that is, many formulas will have a free variable $i$ to which we can pass a tuple (encoded using sets) containing the sequence of individualized atoms. So, instead of assuming that the classes of structures are closed under individualization, we work with the free variable $i$ to which all possible tuples can be passed. In what follows, we will always assume that tuples do not contain duplicates. Moreover, we will freely switch between the “internal” representation of tuples in CPT and the “external” sequence of individualized atoms whenever needed.

For two CPT terms $t$ and $s$ defining tuples, we write $s \subseteq t$ for a more complex CPT term concatenating the tuples given by $s$ and $t$ while discarding duplicates. Here, we also allow $t$ to output the empty set (which is ignored) or a singleton set (whose unique member is appended if not already contained). We write $s_i$ for the term extracting the $i$-th position of the tuple defined by $s$ or the empty set if $i$ is larger than the length of the tuple ($i$ is encoded as a von Neumann ordinal). Clearly, all these terms can be defined in CPT.

We now introduce various notions related to defining isomorphism and canonization. We will show that all of them are equivalent in CPT+WSC. It will be important to not only consider all structures $\mathfrak{A}$ of a given class of structures $\mathcal{K}$, but also to consider all pairs $(\mathfrak{A}, \bar{a})$ for all $\bar{a} \in A^*$. This resembles closure under individualization, as discussed before, but is formally easier. At crucial points we remind the reader that we require closure under individualization in that sense. In what follows, let $L$ be one of the logics CPT or CPT+WSC.

**Definition 4.1 (Definable Isomorphism).** A logic $L$ defines isomorphism for a class of $\tau$-structures $\mathcal{K}$ if there is an $L$-formula $\Phi_{(12)}$, such that, for all (connected) $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$ and $\bar{a} \in A^*$, $\bar{b} \in B^*$, on the disjoint union $\mathfrak{A} \uplus \mathfrak{B}$ it holds that $(\bar{a}, \bar{b}) \in [\Phi_{(12)}]_{\mathfrak{A} \uplus \mathfrak{B}}$ if and only if $(\mathfrak{A}, \bar{a}) \equiv (\mathfrak{B}, \bar{b})$.

In the case of CPT+WSC, we require in the previous definition that $\Phi$ never outputs $\uparrow$. So we can write $(\bar{a}, \bar{b}) \in [\Phi_{(12)}]_{\mathfrak{A} \uplus \mathfrak{B}}$ because we can regard $[\Phi_{(12)}]_{\mathfrak{A} \uplus \mathfrak{B}}$ again as the set of tuples satisfying $\Phi$. In all definitions that follow, we also require without further mentioning that $\uparrow$ never occurs for any input.

**Definition 4.2 (Distinguishable Orbits).** We say that a class of $\tau$-structures $\mathcal{K}$ has $L$-distinguishable $k$-orbits if there is an $L$-formula $\Phi_{(1, x, y)}$ such that for every $\mathfrak{A} \in \mathcal{K}$ and every $\bar{a} \in A^*$ the relation $\leq$ on $k$-tuples defined by $\bar{b} \leq \bar{c}$ if and only if $(\bar{a}, \bar{b}, \bar{c}) \in [\Phi_{(1, x, y)}]$ is a total preorder and its equivalence classes are the $k$-orbits of $(\mathfrak{A}, \bar{a})$.

Note that because $\leq$ is a total preorder and not just some equivalence relation, it defines a total order on the $k$-orbits.

**Definition 4.3 (Complete Invariant).** An L-definable complete invariant of a class of $\tau$-structures $\mathcal{K}$ is an $L$-term $s(i)$ which satisfies the following: $[s]_{\mathfrak{A}}(\bar{a}) = [s]_{\mathfrak{B}}(\bar{b})$ if and only if $(\mathfrak{A}, \bar{a}) \equiv (\mathfrak{B}, \bar{b})$ for every $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$ and every $\bar{a} \in A^*$, $\bar{b} \in B^*$.

**Lemma 4.4.** If there is an $L$-definable complete invariant for a class of $\tau$-structures $\mathcal{K}$, then $\mathcal{K}$ has $L$-distinguishable $k$-orbits for every $k \in \mathbb{N}$.

**Proof Sketch.** Let $\mathfrak{A} \in \mathcal{K}$. We define a total preorder on the $k$-tuples $\bar{u}$, by comparing the output of the complete invariant on $(\mathfrak{A}, \bar{u})$. These outputs are ordered objects (they are necessarily HF($\emptyset$)-sets), so can be ordered in $L$.

**Definition 4.5 (Ready for Individualization).** A class of $\tau$-structures $\mathcal{K}$ is ready for individualization in $L$ if there is an $L$-term $s(i)$ that for every structure $\mathfrak{A} \in \mathcal{K}$ and every $\bar{a} \in A^*$ defines a 1-orbit $O$ of $(\mathfrak{A}, \bar{a})$, that is $O = [s]_{\mathfrak{A}}(\bar{a}) \in \text{orbs}_s((\mathfrak{A}, \bar{a}))$, such that if there is an orbit disjoint with $\bar{a}$ then $O$ is disjoint with $\bar{a}$.

**Definition 4.6 (Canonization).** For a class of $\tau$-structures $\mathcal{K}$, $L$ defines a canonization if there is an $L$-interpretation $\Theta(i)$ mapping $\tau$-structures to $(\tau \uplus \leq)$-structures such that

- $\leq$ is always a total order on the universe,
- the reduct $\Theta(\mathfrak{A}, \bar{a}) \uplus \tau$ satisfies $\Theta(\mathfrak{A}, \bar{a}) \uplus \tau \equiv (\mathfrak{A}, \bar{a})$ for every $\mathfrak{A} \in \mathcal{K}$ and $\bar{a} \in A^*$, and
- $\Theta(\mathfrak{A}, \bar{a}) = \Theta(\mathfrak{B}, \bar{b})$ if and only if $(\mathfrak{A}, \bar{a}) \equiv (\mathfrak{B}, \bar{b})$ for every $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$, $\bar{a} \in A^*$, and $\bar{b} \in B^*$.

We show that the algorithm of [19] is CPT+WSC definable. Intuitively, we iteratively individualize an atom of a non-trivial orbit which is minimal according to an isomorphism-invariant total order on the orbits. We continue this procedure, until we individualized all atoms and thus defined a total order on the atoms. While the order itself is not unique, the isomorphism type of the ordered structure is, because we always choose from orbits. In that way, we obtain the canon by renaming the atoms to numbers. We now define this approach formally: Let $\mathcal{K}$ be a class of $\tau$-structures ready for individualization in CPT+WSC and let $s_{\text{ach}}(\bar{i})$ be a CPT+WSC term defining a 1-orbit with the required properties.

We define for every $\mathfrak{A}$ and $\bar{a} \in A^*$ a set labels$(\mathfrak{A}, \bar{a})$: If all atoms are individualized, i.e., every atom is contained in $\bar{a}$, we set labels$(\mathfrak{A}, \bar{a}) := \{\bar{a}\}$. Otherwise, let $O = [s_{\text{ach}}]_{\mathfrak{A}}(\bar{a})$ be the 1-orbit disjoint with $\bar{a}$ given by $s_{\text{ach}}$. We define

$$\text{labels}(\mathfrak{A}, \bar{a}) := \bigcup_{\bar{u} \in O} \text{labels}(\mathfrak{A}, \bar{u}).$$

For $\bar{b} \in \text{labels}(\mathfrak{A}, \bar{a})$ define $q_{\bar{b}}: A \rightarrow |A|$ via $u \mapsto i$ if and only if $u = b_i$. It is easy to see that labels$(\mathfrak{A}, \bar{a})$ is an $(\mathfrak{A}, \bar{a})$-orbit. Hence, the definition canon$(\mathfrak{A}, (\mathfrak{A}, \bar{a}) := \Theta_{\bar{a}}(\mathfrak{A}, \bar{a})$ is well-defined and independent of $\bar{b} \in \text{labels}(\mathfrak{A}, \bar{a})$ since $q_{\bar{b}}((\mathfrak{A}, \bar{a}) = q_{\bar{c}}((\mathfrak{A}, \bar{a}))$ for every $\bar{b}, \bar{c} \in \text{labels}(\mathfrak{A}, \bar{a})$.

**Lemma 4.7.** It holds that canon$(\mathfrak{A}, \bar{a}) \equiv (\mathfrak{A}, \bar{a})$ and that canon$(\mathfrak{A}, \bar{a}) \equiv \Theta(\mathfrak{A}, \bar{b})$ if and only if $\Theta(\mathfrak{A}, \bar{a}) \equiv (\mathfrak{B}, \bar{b})$.

**Lemma 4.8.** If a class of $\tau$-structures $\mathcal{K}$ is ready for individualization in CPT+WSC, then CPT+WSC defines a canonization for $\mathcal{K}$-structures.

**Proof.** To implement the former approach in CPT+WSC we first need some notation: we introduce a fixed-point operator with deterministic choice similar to the WSC-fixed-point operator, but which can be simulated just with iteration terms (without choice): $\text{DC}^x y_1 \cdot \text{step} \cdot \text{choice} \cdot \text{order}_1$, where $\text{step} \cdot \text{choice}$ and $\text{order}_1$ are CPT+WSC terms. The first terms $\text{step}(x, y)$ and $\text{choice}(x)$ have exactly as in the symmetric choice operator: $\text{step}$ defines a step function and $\text{choice}$ a choice set (of atoms). But the third term $\text{order}_1$ defines a total order on the atoms and resolves the choices.
by picking the minimal one. We define a CPT+WSC interpretation
\( \Theta(i) \) whose set of atoms is \( |A| \) and for every k-ary relation \( E \in \tau \) a formula
\( \Psi_E(t_1, \ldots, t_k) \).

\[
t_{\text{label}}(o, i) := \text{DC}^* xy. (xy, s_{\text{orb}}(x), o)
\]

\[
t_{\text{aut}}(o, i, x, y) := \{(t_{\text{label}}(o, x), t_{\text{label}}(o, y)) \} \quad j \in |\text{Card}(\text{Atoms})|
\]

\[
t_{\text{wit}}(o, i) := \{ t_{\text{aut}}(o, i, x, y) \mid x, y \in s_{\text{orb}}(i) \}
\]

\[
\Psi_E(t_1, \ldots, t_k) := \text{WSC}^* xy. (xy, s_{\text{orb}}(x), t_{\text{wit}}(y, x), \\
R(x_1, \ldots, x_k))
\]

Fix an arbitrary structure \( \mathfrak{A} \in \mathcal{K} \). Set \( A^c \) to be the set of all \( A \)-tuples of length \( |A| \) containing all atoms exactly once.

Claim 1. If \( \bar{b} \in A^c \) then \( \models t_{\text{label}}(o, i) \bar{a} \in \text{labels}(\mathfrak{A}, \bar{a}) \) for every \( \bar{a} \in A^* \).

Claim 2. If \( \bar{b} \in A^c \) then \( \models t_{\text{aut}}(o, i) \bar{a} \) witnesses that
\( \models \{s_{\text{orb}}(i)\} \bar{a} \) is an orbit of (\( \mathfrak{A}, \bar{a} \)).

Claim 3. For every tuple \( \bar{a} \in A^* \) it holds that labels(\( \mathfrak{A}, \bar{a} \) = \n\text{WSC}^* (\{t_{\text{aut}}(o, i) \bar{a} \}, t_{\text{wit}}(\bar{a}, y), \{t_{\text{wit}}(y, \bar{a})\}).

Now let \( \bar{a} \in A^c \) and show that \( \Theta(\mathfrak{A}, \bar{a}) = \text{canon}(\mathfrak{A}, \bar{a}) \). By Claim 3 we have \( (\bar{a}, i_1, \ldots, i_e) \in \Psi_E^{\mathfrak{A}} \) if and only if \( (i_1, \ldots, i_e) \in R^{\mathfrak{G},(\mathfrak{A},\bar{a})} \) for some (and every) \( \bar{b} \in \text{labels}(\mathfrak{A}, \bar{a}) \). This is exactly the definition of canon(\( \mathfrak{A}, \bar{a} \)) and so we obtained a CPT+WSC-definable canonization (Lemma 4.7).

Lemma 4.9 ([17]). If CPT defines isomorphism of a class of binary \( \tau \)-structures \( \mathcal{K} \) (closed under individualization), then there is a CPT term defining a complete invariant for \( \mathcal{K} \).

While this lemma is only for binary structures, it can be applied to arbitrary structures. Every \( \tau \)-structure can be encoded by a binary structure using a CPT-interpretation \( \Theta \) (in fact, an FO-interpretation suffices) such that \( \Theta(\mathfrak{G}) \cong \Theta(\mathfrak{H}) \) if and only if \( \mathfrak{G} \cong \mathfrak{H} \) and given a definable isomorphism test for a class of \( \tau \)-structures \( \mathcal{K} \), we can define an isomorphism test on \( \Theta(\mathcal{K}) \).

Corollary 4.10. If CPT defines isomorphism of a class of \( \tau \)-structures \( \mathcal{K} \), then CPT+WSC defines canonization of \( \mathcal{K} \)-structures and captures PTIME on \( \mathcal{K} \)-structures.

This corollary is asymmetric in the sense that we can convert an isomorphism-defining CPT formula into a CPT+WSC formula defining

\[
\text{canonization. Our goal is to prove the version, which already starts with an isomorphism-defining CPT+WSC formula, stated in the following theorem.}
\]

Theorem 4.11. If CPT+WSC defines isomorphism of a class of binary \( \tau \)-structures \( \mathcal{K} \), then CPT+WSC defines a complete invariant for \( \mathcal{K} \)-structures.

Proving Theorem 4.11 is involved and the proof is deferred to Section 5. Assuming Theorem 4.11 for now, we conclude:

Theorem 4.12. Let \( \mathcal{K} \) be a class of \( \tau \)-structures (closed under individualization). The following are equivalent:

1. \( \mathcal{K} \) is ready for individualization in CPT+WSC.
2. \( \mathcal{K} \) has CPT+WSC-distinguishable 1-orbits.
3. \( \mathcal{K} \) has CPT+WSC-distinguishable k-orbits for every k.
4. CPT+WSC defines isomorphism of \( \mathcal{K} \).
5. CPT+WSC defines a complete invariant for \( \mathcal{K} \).
6. CPT+WSC defines a canonization for \( \mathcal{K} \).

Proof of Theorem 1.1. Canonization is definable in CPT+WSC by Theorem 4.12 for \( \mathcal{K} \) and so by the Immerman-Vardi Theorem [20]
CPT+WSC captures PTIME.

Corollary 4.13. Under the assumption that graph isomorphism is in PTIME, CPT+WSC defines isomorphism on all structures if and only if CPT+WSC captures PTIME.

5 ISOMORPHISM TESTING IN CPT+WSC

The goal of this section is to prove Theorem 4.11. The proof of

Lemma 4.9 in [17] uses the equivalence between CPT and the DeepWL computation model. We extend DeepWL with witnessed symmetric choice. Proofs in the DeepWL model are technical, so we present our results only on a high-level and focus on the differences to [17] necessary due to our new setting. For a more elaborate introduction into DeepWL we refer to [17] and for full proofs to [24].

In the rest of this section, we assume that all structures are binary relational structures. Moreover, we see all relation symbols as binary strings, so we can work with these symbols with Turing machines using a fixed alphabet.

Preliminaries. Let \( \mathfrak{G} \) be a binary \( \tau \)-structure. The inverse of a relation \( E^\mathfrak{G} \) (for some \( E \in \tau \) is \( (E^\mathfrak{G})^{-1} := \{(u, v) \mid (v, u) \in E^\mathfrak{G}\} \). We call \( E \) undirected if \( E^\mathfrak{G} = (E^\mathfrak{G})^{-1} \) and directed otherwise. Let \( \pi \subseteq \tau \) and two atoms \( u, v \in A \) are \( \pi \)-connected if there is a path from \( u \) to \( v \) only using edges contained in a relation in \( \pi \). Similarly, we define \( \pi \)-connected components and strongly \( \pi \)-connected components (SCCs).

A special class of structures are the so-called coherent configurations. Most important, if \( \sigma \)-structure \( \mathfrak{G} \) is a coherent configuration, it possess so-called intersection numbers \( q^\sigma \mapsto \mathbb{N} \): for all \( \sigma \)-relations \( R, S, T \) and every \( (u, v) \in R^\mathfrak{G} \) there are exactly \( q(R, S, T) \) many \( w \in H \) such that \( (u, w) \in S^\mathfrak{G} \) and \( (w, v) \in T^\mathfrak{G} \). The canonical coarsest coherent configuration \( C(\mathfrak{G}) \) refining a \( \tau \)-structure \( \mathfrak{G} \) can be computed with the two-dimensional Weisfeiler-Leaman algorithm. For background on coherent configurations and their use in DeepWL we refer to [5, 17].

Structures with Sets as Vertices. In the following, relational structures in which some “atoms” are obtained as HF-sets of the “old” atoms play an important role. We formalize this as follows: A finite binary \( \tau \)-HF-structure \( \mathfrak{G} \) is a tuple \( (A, M, E^\mathfrak{G}_1, \ldots, E^\mathfrak{G}_k) \), where \( A \) is a set of atoms, \( M \) a finite set of HF(A)-sets, and \( E^\mathfrak{G}_i \subseteq (A \cup M) \). We call \( A \) atoms, and \( V(\mathfrak{G}) := A \cup M \) vertices. In that sense, every \( \tau \)-HF-structure \( \mathfrak{G} \) can be turned into a \( \tau \)-structure \( \mathfrak{G}^\text{flat} \), where the sets in \( M \) become fresh atoms, and every \( \tau \)-structure is also a \( \tau \)-HF-structure, where the set \( M \) is empty.

An automorphism of the \( \tau \)-HF-structure \( \mathfrak{G} \) is a permutation \( \varphi \) of \( A \) such that \( \varphi(M) = M \) and \( (u, v) \in E^\mathfrak{G}_i \) if and only if \( \varphi(u, v) \in E^\mathfrak{G}_i \) for every \( i \in [k] \) and every \( u, v \in V(\mathfrak{G}) \). That is, an automorphism of \( \mathfrak{G} \) has to respect the HF-structure of the vertices. So a \( \tau \)-HF-structure \( \mathfrak{G} \) has potentially fewer automorphisms than the \( \tau \)-structure \( \mathfrak{G}^\text{flat} \). Using this notion of automorphisms, \( \mathfrak{G} \)-orbits and \( (\mathfrak{G}, \bar{a}) \)-orbits (for a tuple \( \bar{a} \) in HF(A)-sets) are defined as before. We set \( C(\mathfrak{G}) := C(\mathfrak{G}^\text{flat}) \).
5.1 DeepWL

We introduce the notion of a DeepWL-algorithm from [17]. A DeepWL-algorithm is a two-tape Turing machine using the alphabet \( \{0, 1\} \) with three special states \( q_{\text{addPair}}, q_{\text{ssc}}, \) and \( q_{\text{create}} \). The first tape is called the work-tape and the second one the interaction-tape. The Turing machine computes on a binary relational \(-HF\)-structure \( \mathcal{A} \), but it has no direct access to it. Instead, the structure is put in the so-called “cloud” which maintains the pair \((\mathcal{A}, C(\mathcal{A}))\). The Turing machine only has access to the algebraic sketch \( D(\mathcal{A}) = (\tau, \sigma, \sqsubseteq, \sigma, q) \), which is written on the interaction tape. Here, \( \sigma \) is the signature of \( C(\mathcal{A}) \). The symbolic subset relation \( \sqsubseteq, \sigma = \{ (R, E) \in \tau \times \sigma \mid R^{C(\mathcal{A})} \subseteq E^{\mathcal{A}} \} \subseteq \tau \times \sigma \) relates a \( \sigma \)-relation \( R \) to the \( \tau \)-relation which is refined by \( \sigma \) (i.e., \( R^{C(\mathcal{A})} \subseteq E^{\mathcal{A}} \)). Finally, \( q \) are the intersection numbers of \( C(\mathcal{A}) \).

In the following, the signature of the \( \mathcal{H}\)-structure \( \mathcal{A} \) in the cloud is (unless stated otherwise) always \( \tau \) and the signature of \( C(\mathcal{A}) \) is \( \sigma \), which is always disjoint from \( \tau \). We call relations \( R \in \tau \) colors and relations \( E \in \tau \) just relations. If \( E \) (respectively \( R \)) is a diagonal relation we identify \( E \) (or \( R \)) with the set \( \{ u \mid (u, u) \in E \} \) and call \( E \) a vertex class (or \( R \) a fiber). For relations we use the letters \( E \) and \( F \), for vertex classes the letters \( C \) and \( D \), for colors the letters \( R, S \), and \( T \), and for fibers the letters \( U \) and \( V \). Although the cloud contains the pair \((\mathcal{A}, C(\mathcal{A}))\), we will just say that \( \mathcal{A} \) is in the cloud and interpret \( \sigma \)-colors \( R \in \mathcal{A} \), i.e., just write \( R^{\mathcal{A}} \) for \( R^{C(\mathcal{A})} \).

With the special states \( q_{\text{addPair}}, q_{\text{ssc}}, \) and \( q_{\text{create}} \) the Turing machine modifies the cloud in an isomorphism-invariant manner by adding vertices. To enter the states \( q_{\text{addPair}} \) and \( q_{\text{ssc}} \), the Turing machine has to write a single relation symbol \( X \in \tau \cup \sigma \) on the interaction-tape. To enter \( q_{\text{create}} \), a set \( \pi \subseteq \sigma \) has to be written on the interaction-tape. We say that the machine executes \( \text{addPair}(X), \quad \text{ssc}(X) \), and \( \text{create}(\pi) \): 

- \( \text{addPair}(X) \): For every \( (u, v) \in X^{\mathcal{A}} \) a fresh vertex \( (u, v) \) (a set-encoded pair) is added to the structure. New relations \( E_{\text{left}} \) and \( E_{\text{right}} \) are added to \( \tau \) containing the pairs \( ((u, v)), (u, v) \) and \( ((u, v)), (u, v) \) respectively.
- \( \text{ssc}(X) \): For every strongly \( X \)-connected component \( c \) a new vertex \( c \) is added (note that \( c \) is itself an \( HF(\mathcal{A}) \))-set). A new relation symbol \( E_{\text{in}} \) is added to \( \tau \) containing the pairs \( (c, u) \) for every \( X \)-SCC \( c \) and every \( u \in c \).
- \( \text{create}(\pi) \): A new relation symbol \( E \) is added to \( \tau \), which is the union of all \( R \in \pi \).

Each of these three operations modify the structure \( \mathcal{A} \). After that, the coherent configuration \( C(\mathcal{A}) \) is recomputed and the new algebraic sketch \( D(\mathcal{A}) \) is written on the interaction-tape. Then the Turing machine continues. A DeepWL-algorithm accepts \( \mathcal{A} \) if the work-tape points to a 1 when the Turing machine halts and rejects otherwise.

We differ at various places from the original definition of a DeepWL-algorithm given in [17]. Nevertheless, our definition has the same expressiveness with respect to deciding properties of structures using DeepWL-algorithms. Our operations cannot “forget” structural information of the structure in the cloud and allow us to establish the following lemma. It justifies computing \( C(\mathcal{A}) \) on the \( \tau \)-structure \( \mathcal{A}^{\text{flat}} \) and that DeepWL does not need to access the \( \mathcal{H} \)-structure of \( \mathcal{A} \) if the original input was a \( \tau \)-\((\text{non-HF})\)-structure.

**Lemma 5.1.** Let \( \mathcal{A}_0 \) be a \( \tau \)-structure and let \( \mathcal{A} \) be a \( \tau \)-HF-structure obtained by a DeepWL-algorithm in the cloud on input \( \mathcal{A}_0 \). Then for every automorphism \( \phi \in \text{Aut}(\mathcal{A}^{\text{flat}}) \) it holds that \( \phi(\mathcal{A}) = \mathcal{A} \).

The proof uses that the relations \( E_{\text{left}}, E_{\text{right}}, \) and \( E_{\text{in}} \) encode the structure of the HF-sets in the relations, which cannot be “forgotten” at a later point. This lemma is in particular important when we extend DeepWL with a witnessed choice operator: to compute (HF-set respecting) orbits of \( \mathcal{A} \) it suffices to consider \( C(\mathcal{A}^{\text{flat}}) \).

5.2 DeepWL with Witnessed Symmetric Choices

We add witnessed symmetric choices to the DeepWL computation model. To do so, we need two different notions: we compose DeepWL+WSC machines to DeepWL+WSC algorithms. DeepWL+WSC machines in some sense will play the role of terms in CFP+WSC and DeepWL+WSC algorithms the role of WSC-fixed-point operators. A DeepWL+WSC machine just can make an arbitrary choice, but composed in DeepWL+WSC algorithms there must be witnessing automorphisms. We need to nest DeepWL+WSC algorithms (in contrast to plain DeepWL-algorithms), because – as seen earlier – operators with witnessed symmetric choice have to syntactically return an isomorphism-invariant result (again true or false). We now formally define these objects.

A DeepWL+WSC-machine \( M \) is a DeepWL-algorithm, in which the Turing machine has two additional states \( q_{\text{choice}} \) and \( q_{\text{refine}} \). To enter \( q_{\text{choice}} \), \( M \) has to write a relation symbol \( X \in \tau \cup \sigma \) on the interaction-tape. To enter \( q_{\text{refine}} \) it has to write a relation symbol \( X \in \tau \cup \sigma \) and a number on the interaction-tape. We say that the machine \( M \) executes \( \text{choice}(X) \) and \( \text{refine}(X, i) \). The DeepWL+WSC-machine \( M \) is choice-free, if it never (syntactically) enters \( q_{\text{choice}} \). That is, \( q_{\text{choice}} \) is not in the range of the transition function of the underlying Turing machine of \( M \).

**Definition 5.2.** A DeepWL+WSC-algorithm \( M \) is a tuple \( (M^{\text{out}}, M^{\text{wit}}, M_1, \ldots, M_r) \) of a DeepWL+WSC-machine \( M^{\text{out}} \), a choice-free DeepWL+WSC-machine \( M^{\text{wit}} \), and (a possibly empty) sequence of DeepWL+WSC-machines \( M_1, \ldots, M_r \). The machine \( M^{\text{out}} \) is called the output machine and the machine \( M^{\text{wit}} \) is called the witness machine.

Note that the prior definition is an inductive definition. We discuss informally how a DeepWL+WSC-algorithm \( M = (M^{\text{out}}, M^{\text{wit}}, M_1, \ldots, M_r) \) is executed. We start with the two DeepWL+WSC-machines \( M^{\text{out}} \) and \( M^{\text{wit}} \).

If a DeepWL+WSC-machine \( M \in \{M^{\text{out}}, M^{\text{wit}}\} \) executes \( \text{refine}(X, j) \) and \( j > \ell \), then \( M \) just continues. Otherwise, the DeepWL+WSC-algorithm \( M_1 \) is used to refine the relation \( X \): Let \( \mathcal{A} \) be the content of the cloud of \( M \) when it executes the \( \text{refine} \)-operation. If \( X \) is directed, then the algorithm \( M_1 \) is executed on \( (\mathcal{A}, u, v) \) (and \( u, v \) are individualized by putting them in singleton vertex classes) for every \( (u, v) \in X^{\mathcal{A}} \). Otherwise, \( X \) is undirected and for every \( (u, v) \in X^{\mathcal{A}} \) the algorithm \( M_1 \) is executed on \( (\mathcal{A}, \{u, v\}) \) (the undirected edge is individualized by a new vertex class only containing \( u, v \)). The algorithm \( M_1 \) modifies its own cloud independently of the cloud of \( M \). If \( M_1 \) accepts all \( (\mathcal{A}, u, v) \) for all \( (u, v) \in X^{\mathcal{A}} \) (respectively all \( (\mathcal{A}, \{u, v\}) \) for all \( (u, v) \in X^{\mathcal{A}} \)), then nothing happens. Otherwise, a new relation \( E' \) is added to the cloud.
of $M$, where $E'$ consists of all $(u, v) \in X^R$ (respectively $\{u, v\} \in X^R$), for which $M_j$ accepts the input. Note that the content of the cloud remains an HF-structure with the same set of atoms.

If $M$ executes $\text{choice}(X)$ and $X$ is a directed relation, an arbitrary $(u, v) \in X^R$ is individualized and $M$ continues. If otherwise $X$ is undirected, then an undirected edge $\{u, v\} \in X^R$ is individualized as in the refine-operation.

Treating directed and undirected relations differently in the refine- and the choice-operation seems odd at first. For general DeepWL+WSC-algorithms such a distinction is not necessary because it can be simulated. We introduce the notion of a normalized DeepWL+WSC-algorithm in Section 5.4, which puts additional restrictions on e.g., addPair operations. Here the precise semantics of the refine- and choice-operations matter. It is crucial to prove Lemma 5.13.

The algebraic sketch is recomputed and written on the interaction-tape when refine$(X, j)$ and choice$(X)$ modify the structure in the cloud. The machine $M$ accepts the input, if the symbol under the head on the work-tape is a 1 when $M$ halts and rejects otherwise.

We now turn to the DeepWL+WSC-algorithm $M$. To execute the algorithm $M$, the output machine $M^{\text{out}}$ is executed. Let $\mathcal{A}_0$ be the input to $M^{\text{out}}$ and $F$ be the content of the cloud when $M^{\text{out}}$ halts. For every choice-operation executed by $M^{\text{out}}$, the witnessing machine $M^{\text{wit}}$ is executed. Let $k$ be a number, $\text{choice}(X_i)$ be the $i$-th executed choice-operation (for some $X_i$ in the current signature), and $\mathcal{A}_i$ be the content of the cloud, when the $i$-th choice-operation is executed for every $i \in [k]$. For the $k$-th choice-operation the machine $M^{\text{wit}}$ has to provide automorphisms witnessing that $X^R_i$ is an $\langle \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k \rangle$-orbit. This is similar to the WSC-fixed-point operator where all intermediate steps of the fixed-point computation have to be fixed. Because all HF-structures have the same set of atoms $A_0$, the notion of an $\langle \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k \rangle$-orbit is well-defined. The refine-operation ensures similarly to output formulas that only isomorphism-invariant results (accept or reject) are propagated.

The input of $M^{\text{wit}}$ is the labeled union $\mathcal{A} \sqcup \mathcal{A}_k$, which is the union $\mathcal{A} \cup \mathcal{A}_k$ equipped with two fresh relation symbols $E_1$ and $E_2$ labeling the vertices of $\mathcal{A}$ and $\mathcal{A}_k$, i.e., $E_1^{\mathcal{A} \sqcup \mathcal{A}_k} = \mathcal{V}(\mathcal{A})$ and $E_2^{\mathcal{A} \sqcup \mathcal{A}_k} = \mathcal{V}(\mathcal{A}_k)$. So $M^{\text{wit}}$ can reconstruct $\mathcal{A}$ and $\mathcal{A}_k$ and can also determine how $\mathcal{A}$ and $\mathcal{A}_k$ relate to each other since common vertices are merged in the union. When the witnessing machine $M^{\text{wit}}$ halts, it has to write a relation symbol on the interaction-tape. The relation has to encode the set of witnessing automorphisms (details omitted).

If all choices are successfully witnessed, $M$ accepts the input if $M^{\text{out}}$ accepts the input and rejects otherwise. If some choice could not be witnessed, we abort the computation and output $\dagger$. If an executed subalgorithm $M_j$ outputs $\dagger$, $M$ also outputs $\dagger$. We also say that $M$ fails.

The semantics of a DeepWL+WSC-algorithm can be defined formally using the WSC-operator from Section 3.1. The following lemma follows from Corollaries 3.4 and 3.5.

**Lemma 5.3.** For every DeepWL+WSC-algorithm $M$, the class of structures accepted by $M$ is isomorphism-closed. The algorithm $M$ accepts (respectively rejects or fails) independent of the choices made in the execution of the output machine.

A DeepWL+WSC-algorithm $M$ decides a property $P$ of a class of $\tau$-structures $\mathcal{K}$, if $M$ accepts $\mathcal{A} \in \mathcal{K}$ if $\mathcal{A}$ satisfies $P$ and rejects otherwise (and in particular never fails). It computes a function $f : \mathcal{K} \rightarrow \{0, 1\}^\ast$, if $f(\mathcal{A})$ is written onto the tape when $M$ halts on input $\mathcal{A}$. The runtime of $M$ is defined similarly to [17]. A DeepWL+WSC-algorithm runs in polynomial time, if there exists a polynomial $p(n)$, such that $p(|A_0|)$ bounds the runtime on input $\mathcal{A}_0$.

### 5.3 From CPT+WSC to DeepWL+WSC

CPT+WSC formulas can be translated into DeepWL+WSC-algorithms. The translation is based on the translation of CPT into interpretation logic in [13] and the translation of interpretation logic in DeepWL in [17].

**Lemma 5.4.** If a property $P$ is CPT+WSC definable, then there is a polynomial time DeepWL+WSC-algorithm deciding $P$.

### 5.4 Normalized DeepWL+WSC

Now we are interested in DeepWL+WSC-algorithms deciding isomorphism, that is, in DeepWL+WSC-algorithms computing on a disjoint union of two structures. We follow the idea of [17] and show that it suffices never to “mix” vertices of the two components in the cloud. However, we have to differ from their construction at many points and some of our changes are crucial in the presence of choices.

We assume that in the input structure $\mathcal{A}_1 \sqcup \mathcal{A}_2$ each $\mathcal{A}_i$ consists of a single connected component. In the following, we call the $\mathcal{A}_i$ just components. During the execution of a DeepWL+WSC-machine on input $\mathcal{A}_1 \sqcup \mathcal{A}_2$ a vertex $w$ belongs to $\mathcal{A}_i$, if $w \in \mathcal{V}(\mathcal{A}_i)$, that is, $w \in \mathcal{V}(\mathcal{A}_i \sqcup \mathcal{A}_k)$, where $u$ belongs to $\mathcal{A}_j$, or $w$ is obtained as a vertex for an SCC during a scc execution, where $c$ only contains vertices belonging to $\mathcal{A}_j$.

Let $\mathcal{A}$ be the current content of the cloud and set $V_i(\mathcal{A}) := \{ u \in \mathcal{V}(\mathcal{A}) \mid u \text{ belongs to } \mathcal{A}_i \}$. The vertices $V_1(\mathcal{A}) \cup V_2(\mathcal{A})$ and edges $V_1(\mathcal{A})^2 \cup V_2(\mathcal{A})^2$ are called plain and the edges $E_{\text{cross}}(\mathcal{A}) := V_1(\mathcal{A}) \times V_2(\mathcal{A}) \cup V_2(\mathcal{A}) \times V_1(\mathcal{A})$ are called crossing. A relation or color is called plain (respectively crossing) if it only contains plain (respectively crossing) edges.

**Definition 5.5 (Normalized DeepWL+WSC).** A structure $\mathcal{A}$ obtained from a structure $\mathcal{A}_1 \sqcup \mathcal{A}_2$ by a DeepWL+WSC-algorithm is called normalized if every vertex of $\mathcal{A}$ is plain and $\mathcal{A}$ consists of the two connected components $\mathcal{A}_1[V_1(\mathcal{A})]$ and $\mathcal{A}_2[V_2(\mathcal{A})]$. A DeepWL+WSC-machine is normalized if on input $\mathcal{A}_1 \sqcup \mathcal{A}_2$ the content of the cloud $\mathcal{A}$ at any time is normalized. A DeepWL+WSC-algorithm is normalized, if all contained DeepWL+WSC-machines are normalized.

It turns out that crossing colors are just “direct products” of plain fibers and that the algebraic sketch of a normalized structure is given by the sketches of its components.

**Lemma 5.6 (Lemma 8 [17]).** For every normalized structure $\mathcal{A} = \mathcal{A}_1 \sqcup \mathcal{A}_2$ and for every crossing color $R$ there are two plain fibers $U$ and $V$ such that $R^R = (U^R \times V^R) \cap E_{\text{cross}}(\mathcal{A})$. The two sketches $D(\mathcal{A}_1) \cap D(\mathcal{A}_2)$ determine $D(\mathcal{A})$ and in particular satisfy $D(\mathcal{A}) = D(\mathcal{A}_1)[\sigma_1, V_1(\mathcal{A})]$ (where $\sigma_1 \subseteq \sigma$ is the set of relation symbols containing at least one pair of $V_1(\mathcal{A})^2$).
We refer with $\in [i, j]$ suffices to compute the algebraic sketch because $\tau$ are crossing vertices in $A$ only contains crossing vertices. A vertex class or fiber is called crossing, if it are defining an HF-structure). A relation plan $\Omega$ for $A$ is a pair $\Omega = (\omega, \kappa)$, where $\omega$ is a finite set of vertex plans and $\kappa$ is a finite set of relation plans where each relation plan defines a different relation symbol.

A vertex plan $(C, D)$ says that for every $u \in C^\Pi$ and $v \in D^\Pi$ in different components the vertex $\{u, v\}$ is added (recall that we are defining an HF-structure). A relation plan $(E, \{(F_{1, i}, F_{2, i}) \mid i \in [k]\})$ specifies a relation $E$ between the created vertices: a pair $\{u, v\}$, $\{u', v'\}$ is contained in $E^\Omega$ if and only if there is some $i \in [k]$ such that the corresponding plain edges satisfy $(u, u') \in F_{1, i}^\Pi$ and $(v, v') \in F_{2, i}^\Pi$. Lastly, a special relation $E_p$ relating every created vertex $\{u, v\}$ to the original vertices $u$ and $v$ is added.

The added vertices $\{u, v\}$ are called crossing. The set of all crossing vertices is $V_{\text{cross}}(\Pi)$. We call edges $(w, w') \in V_{\text{cross}}(\Pi)^2$ inter-crossing. A relation or color is inter-crossing if it only contains plain inter-crossing edges. A vertex class or fiber is called crossing, if it only contains crossing vertices.

In the following, we are interested only in the substructure of $\Omega_{\text{cross}}(\Pi)$ induced by the crossing vertices. But for HF-structures this is ill-defined because the crossing vertices do not contain the atoms of $\Pi$. So we turn to the non-HF-structure $\Omega_{\text{cross}}(\Pi)^{\text{flat}}$ and define

$$\Omega_{\text{cross}}(\Pi) := \Omega(\Pi)^{\text{flat}}[V_{\text{cross}}(\Omega(\Pi)^{\text{flat}})].$$

We refer with $V_{\text{cross}}(\Omega(\Pi)^{\text{flat}})$ to the set of atoms in $\Omega(\Pi)^{\text{flat}}$ which are crossing vertices in $\Omega(\Pi)$. However, using the non-HF-structure suffices to compute the algebraic sketch because $D(\Pi)$ was defined via $C(\Pi) = C(\Pi)^{\text{flat}}$.

The structures obtained from building plans differ at some crucial points from the almost normalized structures of [17]:

- Our notion of crossing vertices is defined only for building plans and not for general DeepWL-algorithms (and is for undirected crossing edges).
- Our relation $E_p$ only contains the pairs $\{(u, v), u\}$ and $\{(u, v), v\}$ and so does not distinguish between $u$ and $v$. So a choice-operation on crossing vertices does not necessarily distinguish the two components.
- The relation $E_p$ assigns to every crossing vertex exactly one plain vertex in each component. This ensures that choice sets can still be witnessed.

Using results from [17], we can show that the algebraic sketch of $\Omega(\Pi)$ can be computed from the sketch of $\Pi$ using a DeepWL-algorithm (even without the extension for choices).

**Lemma 5.8.** There is a DeepWL-algorithm that for every normalized structure $\Pi$ and every building plan $\Omega = (\omega, \kappa)$ for $\Pi$ computes $D(\Omega(\Pi))$ in polynomial time.

Now, the goal is to show that an arbitrary DeepWL+WSC algorithm can be simulated by a normalized DeepWL+WSC algorithm which uses building plans to describe a potentially non-normalized structure. To do so, we have to solve two issues not present in [17].

**The Direct Product Property and refine.** The first issue is that refine-operations can violate the "direct product" property of Lemma 5.6. We give an illustrating example: Assume we are given a normalized structure $\Pi = A_1 \uplus A_2$, both components of the same size, where all atoms are in the same fiber and consequently all crossing edges are in the same color $R$. Further, assume that some DeepWL+WSC-algorithm can linearly order the atoms in each component. Then we can define an algorithm $M$ which, for $(u, v) \subseteq R^2$, accepts $(A_1 \uplus A_2, (u, v))$ if and only if $u$ and $v$ are the $i$-th vertex in their component. If we refine $M$ with $M$, a relation $E$ containing a perfect matching between the components is added. But the information, that the atoms can be ordered, is not "returned". The result is again a coherent configuration without refined fibers and so the "direct product" property is violated. A building plan cannot represent the resulting structure. To fix this, we can instead distinguish the atoms before executing the refine-operation avoiding the problem.

This strategy works in general using a suitable notion of an internal run $\text{run}(M, \Pi)$ of a DeepWL+WSC-machine on a structure $\Pi$ in the cloud (similar to the notion in [17]). Intuitively, the internal run $\text{run}(M, \Pi)$ is the sequence of visited states, contents of the two tapes and algebraic sketches of the structure in the cloud. For refine-operations, the internal runs of the algorithm used to refine the sketch are also included.

One then shows that for a crossing color $R^\Pi = (U^\Pi \times V^\Pi) \cap E_{\text{cross}}(\Pi)$ and every $(u, v), (u', v') \subseteq R^\Pi$ such that $M$ accepts $(\Pi, (u, v))$, and $M$ does not accept $((\Pi, (u, v'))$ we can use the internal run of $M$ to refine the fibers:

$$\text{run}(M, (\Pi, (u, w))) \mid w \in U \}$$

Note that the two sets only depend on $v$ and $v'$, so on a single vertex, but not on a crossing edge. These sets can be computed by another DeepWL+WSC algorithm and used to refine the fibers. Then the crossing edges can be refined using $M$ such that the "direct product" property is maintained (details omitted here).

**SCCs of Inter-Crossing Colors.** The second issue comes from the scc-operation. Our different use of the relation $E_p$ forces us to simulate the scc-operation differently. While in [17] the similar contract-operation can be freely used on almost normalized structures, we need to consider the SCCs in coherent configurations: we have to show that every $R$-SCC of an inter-crossing color $R$ is actually determined by the SCCs of the corresponding plain colors of $R$. 

So executing $\text{addPair}$ for a crossing relation in some sense does not provide more information. This observation is one key to show that for every DeepWL+WSC algorithm there is an equivalent normalized one. Almost normalized structures are used as an intermediate step in [17]. We follow a different approach and use so-called building plans.
Lemma 5.9. Let $\mathcal{A}$ be a normalized structure, $\Omega$ be a building plan for $\mathcal{A}$, $R \in \sigma$ be an inter-crossing color of $\Omega(\mathcal{A})$ such that $R^\mathcal{A} \subseteq (U^\mathcal{A})^2$ for some fiber $U \in \sigma$, and and $S, T \in \sigma$ be the corresponding plain colors of $R$. If $c$ is an $R-$SCC, then $\{u, v\} \in c \cap \Omega(\mathcal{A})$ is an $(S, T)$-SCC for every $i \in [2]$.

Recall that an $E-$SCC $c$ is a set of crossing vertices, that is every vertex in $c$ is of the form $(u, v)$. In the next step, we show that we can actually simulate SCC-operations on building plans:

Lemma 5.10. There is a polynomial time DeepWL+WSC-algorithm that given the normalized structure $\mathcal{A}$, a building plan $\Omega$ for $\mathcal{A}$, and an inter-crossing color $R \in \sigma$ of $\Omega(\mathcal{A})$ halts with the normalized structure $\mathcal{B}$ in the cloud and a building plan $\Omega_R$ for $\mathcal{B}$ written on the tape satisfying $\Omega^\mathcal{B} = \Omega^\mathcal{A}_R$, where $\Omega^\mathcal{A}_R$ is the structure obtained by executing SCC($R$) on $\Omega^\mathcal{A}$.

Simulation. Now we are able to describe how an arbitrary DeepWL+WSC algorithm is simulated by a normalized one. Conceptually, to simulate a structure $\mathcal{A}$ we maintain a normalized structure $\mathcal{B}$ and a building plan $\Omega$ for $\mathcal{A}$, and an inter-crossing color $R \in \sigma$ of $\Omega(\mathcal{A})$ halts with the normalized structure $\mathcal{B}$ in the cloud and a building plan $\Omega_R$ for $\mathcal{B}$ written on the tape satisfying $\Omega^\mathcal{B} = \Omega^\mathcal{A}_R$, where $\Omega^\mathcal{A}_R$ is the structure obtained by executing SCC($R$) on $\Omega^\mathcal{A}$.

Corollary 5.14. If isomorphism of a class of binary $\tau$-structures $\mathcal{K}$ is decidable by a polynomial time DeepWL+WSC-algorithm, then isomorphism on $\mathcal{K}$ is decidable by a normalized polynomial time DeepWL+WSC-algorithm.

Theorem 5.15. There is for a class of binary $\tau$-structures $\mathcal{K}$ a polynomial time DeepWL+WSC-algorithm deciding isomorphism if and only if there is a polynomial time DeepWL+WSC-algorithm computing a complete invariant for $\mathcal{K}$.

The theorem is proven similar to the one for DeepWL [17]. The internal run of the DeepWL+WSC-algorithm on input $\mathcal{A} \cup \mathcal{B}$ turns out to be the complete invariant for $\mathcal{A}$.

5.5 From DeepWL+WSC to CPT+WSC

To prove Theorem 4.11, it remains to show that CPT+WSC can simulate polynomial time DeepWL+WSC-algorithms.

Lemma 5.16. If a function $f$ or property $P$ is computable by a polynomial time DeepWL+WSC-algorithm, then $f$ or $P$ is CPT+WSC definable.

Here it is crucial that automorphisms always respect the vertices in the cloud as HF-sets because in CPT+WSC this always has to be done, too. We can finally prove Theorem 4.11.

Proof of Theorem 4.11. The claim follows from Lemma 5.4, Theorem 5.15, and Lemma 5.16. □

6. The CFI-Query

IFP was shown not to capture Ptime by Cai, Fürer, and Immerman [3] using the so-called CFI-graphs. These graphs come with the problem to decide whether a given CFI-graph is even or not. This is called the CFI-query. Already defining restricted versions of the CFI-query is rather difficult in CPT: the best current result is that the CFI-query for base graphs of logarithmic color class size or base graphs with linear maximal degree is CPT-definable [28]. With witnessed symmetric choice defining the CFI-query becomes easier.

We use the definition of CFI-graphs given in [3]: starting from a connected and simple base graph $G = (V, E)$, every vertex $u$ is replaced by a gadget with a pair of outer vertices for every incident base edge $e = (u, v) \in E$ and some inner vertices, ensuring that every automorphism of the gadget swaps exactly an even number
of these pairs. According to a function $g: E \rightarrow \Gamma_2$ these vertex pairs are connected for adjacent vertices. In that way we obtain the graph $\text{CFI}(G,g)$. The CFI-query is to define whether $\sum g = 0$. If the base graph is colored (i.e. a structure $G = (V,E,\leq)$ equipped with a total preorder), then the gadgets are colored according to their base vertices. The color class size of $G$ is the size of its largest $\leq$-equivalence class. For a class of base graphs $\mathcal{K}$ the class of all CFI graphs over $\mathcal{K}$ is $\text{CFI}(\mathcal{K})$.

Let $G = (V,E,\leq)$ be a base graph and $g: E \rightarrow \Gamma_2$. Automorphisms of $G$ translate to automorphisms of $\text{CFI}(G,g)$ (provided $G$ has a vertex of degree at least 3). These automorphisms respect gadgets: every automorphism of $\text{CFI}(G,g)$ is composed of an automorphism of the base graph $G$ and a "CFI-automorphism" given by the CFI-construction.

**Lemma 6.1.** Let $\mathcal{K}$ be a class of (colored) base graphs with CPT-distinguishable 2-orbits. Then $\text{CFI}(\mathcal{K})$ is ready for individualization in CPT.

Now CFI-graphs can be canonicized using Theorem 4.12 and thus the CFI-query can be defined. Note that we never constructed automorphisms of CFI-graphs explicitly.

**Corollary 6.2.** For every class of (colored) base graphs $\mathcal{K}$ with CPT-distinguishable orbits, CPT-WSC defines the CFI-query.

We finally show that Corollary 6.2 covers graph classes for which it is not known that the CFI-query is CPT-definable.

**Corollary 6.3.** There is a class of (colored) base graphs $\mathcal{K} = \{G_n \mid n \in \mathbb{N}\}$, such that CPT-WSC defines the CFI-query for $\mathcal{K}$ and for every $n \in \mathbb{N}$ the graph $G_n$ is $O(\sqrt{|G_n|})$-regular and every color class of $G_n$ has size $\Omega(\sqrt{|G_n|})$.

**Proof Sketch.** We define $G_n$ as follows: start with $n$ disjoint cliques $K_1, \ldots, K_n$ of size $n$. Then connect every $K_i$ to $K_{i+1}$ (and $K_n$ to $K_1$) with a complete bipartite graph. Finally, color the graph such that each $K_i$ is a color class. Then $\mathcal{K}$ has CPT-definable 2-orbits and satisfies the claim by Corollary 6.2.

### 7 DISCUSSION

We extended CPT with a witnessed symmetric choice operator and obtained the logic CPT-WSC. We proved that defining isomorphism in CPT-WSC is equivalent to defining canonization. A crucial point was to extend the DeepWL computing model to show that a CPT-WSC-definable isomorphism test yields a CPT-WSC-definable complete invariant. Thereby, CPT-WSC can be viewed as a simplification step in the quest for a logic capturing PTIME as now only defining isomorphism suffices to apply the Immerman-Vardi theorem.

To turn a complete invariant into a canonization within CPT-WSC we used the canonization algorithm of Gurevich. To implement it in CPT-WSC, we have to extend it to provide witnessing automorphisms. To do so, we needed to give the witnessing terms the defined fixed-points as input. This is different in other extensions of first order logic with symmetric choice [7, 11]. We require that choice sets are orbits when respecting all previous steps. It appears that this only matters if previous choices are actively forgotten. But how could forgetting these be beneficial? We are not sure whether the modification changes the expressiveness of the logic.

Another question is the relation of CPT+WSC to other logics. Is CPT+WSC more expressive than CPT? Do nested WSC-fixed-point operators increase the expressiveness of CPT+WSC? In [7] it is proven that for fixed-point logic extended with (unwitnessed) symmetric choice, the ability to nest increases expressiveness. We should remark that any positive answer to our questions separates CPT from PTIME and hence all questions might be difficult to answer.

Finally, extending DeepWL with witnessed symmetric choice turned out to be extremely tedious. While proofs for DeepWL without choice are already complicated [17] for our extensions the proofs got even more involved. We would like to see more elegant techniques to prove Theorem 4.11 for CPT-WSC (or even for CPT).

**REFERENCES**

[1] Andreas Blass, Yuri Gurevich, and Sabaron Shelah. 1999. Choiceless Polynomial Time. Ann. Pure Appl. Logic 100, 1-3 (1999), 141–187. https://doi.org/10.1016/S0168-0072(99)00005-6

[2] Andreas Blass, Yuri Gurevich, and Sabaron Shelah. 2002. On Polynomial Time Computation over Unordered Structures. J. Symbol. Log. 67, 3 (2002), 1093–1125. https://doi.org/10.2178/jsl/1190150152

[3] Jin-yi Cai, Martin Furer, and Neil Immerman. 1992. An optimal lower bound on the number of variables for graph identification. Combinatorica 12, 4 (1992), 389–410. https://doi.org/10.1007/BF01930532

[4] Ashok K. Chandra and David Harel. 1982. Structure and Complexity of Relational Queries. J. Comput. Syst. Sci. 25, 1 (1982), 99–128. https://doi.org/10.1016/0022-0000(82)90012-5

[5] Gang Chen and Ilia Ponomarenko. 2019. Lectures on coherent configurations. Central China Normal University Press, Wuhan. A draft is available at www.pdm.ras.ru/~inp/ccNOTES.pdf.

[6] Anuj Dawar, Eric Gradel, and Moritz Lichter. 2021. Limitations of the Invertible-Map Equivalences. CoRR abs/2109.07218 (2021). arXiv:2109.07218 https://arxiv.org/abs/2109.07218

[7] Anuj Dawar and David Richerby. 2003. A Fixed-Point Logic with Symmetric Choice. In: Computer Science Logic, 17th International Workshop, CSL 2003, 12th Annual Conference of the EACSL, and 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, August 25-30, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2803). Matthias Baaz and Johann A. Makowksy (Eds.). Springer, 169–182. https://doi.org/10.1007/978-3-540-45220-1_16

[8] Anuj Dawar and David Richerby. 2005. Fixed-point Logics with Nondeterministic Choice. J. Log. Comput. 15, 3 (2005), 503–530. https://doi.org/10.1093/logcom/exc430

[9] Anuj Dawar, David Richerby, and Benjamin Rossman. 2008. Choiceless Polynomial time, counting and the Cai-Furer-Immerman graphs. Ann. Pure Appl. Logic 152, 1-3 (2008), 31–50. https://doi.org/10.1016/j.apal.2007.11.011

[10] Ronald Fagin. 1974. Generalized first-order spectra and polynomial-time recognizable sets. In Complexity of computation (Proc. SIAM-AMS Sympos. Appl. Math., New York, 1973), 43–73. SIAM–AMS Proc., Vol. VII.

[11] Map Equivalences. F. Gire and H. Khahn Hoang. 1998. An Extension of Fixpoint Logic with a Symmetry-Based Choice Construct. Inf. Comput. 144, 1 (1998), 40–65. https://doi.org/10.1006/inco.1998.2712

[12] Erich Gradel and Martin Grohe. 2015. Is Polynomial Time Choiceless?. In Fields of Logic and Computation II - Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday (Lecture Notes in Computer Science, Vol. 9300). Lev D. Beklemishev, Andreas Blass, Nachum Dershowitz, Bernd Finkbeiner, and Wolfram Schulte (Eds.). Springer, 193–209. https://doi.org/10.1007/978-3-319-23534-9_11

[13] Erich Gradel, Wied Pakusa, Svenja Schalthöfer, and Lukas Kaiser. 2015. Characterising Choiceless Polynomial Time with First-Order Interpretations. In 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015. IEEE Computer Society, 677–688. https://doi.org/10.1109/LICS.2015.68

[14] Martin Grohe. 2008. The Quest for a Logic Capturing PTIME. In Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA. IEEE Computer Society, 267–271. https://doi.org/10.1109/LICS.2008.11

[15] Martin Grohe. 2017. Descriptive Complexity, Canonization, and Definable Graph Structure Theory. Cambridge University Press.
