

Dynamic Modeling in Inductive Inference

John Case, **Timo Kötzing**
University of Delaware

October 14, 2008

Motivation

- ▶ In cognitive science, **theory of mind** refers to ones having a model (or models) of another's thoughts, emotions, and perspectives including those different from ones own.
- ▶ Ideally, one might want to have a program (or programs) **generating the behavior** of the other's elicited behavior.
- ▶ But the behavior presented by the other would, in reality, be all and only that **resulting from the interaction between oneself and the other**.
- ▶ While one is attempting to **learn** program(s) for the other, a technique to employ is to pass on a sequence of remarks such as, "I think you are like ...", (where ... might be a program), and, then, attend to the resultantly elicited sequence of reactions of the other as one formulates further programs/models of the other.
- ▶ This active interplay of learner and learnee leads to a **new** learning paradigm, called **Dynamic Modeling**.

Motivation

- ▶ In cognitive science, **theory of mind** refers to ones having a model (or models) of another's thoughts, emotions, and perspectives including those different from ones own.
- ▶ Ideally, one might want to have a program (or programs) **generating the behavior** of the other's elicited behavior.
- ▶ But the behavior presented by the other would, in reality, be all and only that **resulting from the interaction between oneself and the other**.
- ▶ While one is attempting to **learn** program(s) for the other, a technique to employ is to pass on a sequence of remarks such as, "I think you are like ...", (where ... might be a program), and, then, attend to the resultantly elicited sequence of reactions of the other as one formulates further programs/models of the other.
- ▶ This active interplay of learner and learnee leads to a **new** learning paradigm, called **Dynamic Modeling**.

Motivation

- ▶ In cognitive science, **theory of mind** refers to ones having a model (or models) of another's thoughts, emotions, and perspectives including those different from ones own.
- ▶ Ideally, one might want to have a program (or programs) **generating the behavior** of the other's elicited behavior.
- ▶ But the behavior presented by the other would, in reality, be all and only that **resulting from the interaction between oneself and the other**.
- ▶ While one is attempting to **learn** program(s) for the other, a technique to employ is to pass on a sequence of remarks such as, "I think you are like ...", (where ... might be a program), and, then, attend to the resultantly elicited sequence of reactions of the other as one formulates further programs/models of the other.
- ▶ This active interplay of learner and learnee leads to a **new** learning paradigm, called **Dynamic Modeling**.

Motivation

- ▶ In cognitive science, **theory of mind** refers to ones having a model (or models) of another's thoughts, emotions, and perspectives including those different from ones own.
- ▶ Ideally, one might want to have a program (or programs) **generating the behavior** of the other's elicited behavior.
- ▶ But the behavior presented by the other would, in reality, be all and only that **resulting from the interaction between oneself and the other**.
- ▶ While one is attempting to **learn** program(s) for the other, a technique to employ is to pass on a sequence of remarks such as, "I think you are like ...", (where ... might be a program), and, then, attend to the resultantly elicited sequence of reactions of the other as one formulates further programs/models of the other.
- ▶ This active interplay of learner and learnee leads to a **new** learning paradigm, called **Dynamic Modeling**.

Motivation

- ▶ In cognitive science, **theory of mind** refers to ones having a model (or models) of another's thoughts, emotions, and perspectives including those different from ones own.
- ▶ Ideally, one might want to have a program (or programs) **generating the behavior** of the other's elicited behavior.
- ▶ But the behavior presented by the other would, in reality, be all and only that **resulting from the interaction between oneself and the other**.
- ▶ While one is attempting to **learn** program(s) for the other, a technique to employ is to pass on a sequence of remarks such as, "I think you are like ...", (where ... might be a program), and, then, attend to the resultantly elicited sequence of reactions of the other as one formulates further programs/models of the other.
- ▶ This active interplay of learner and learnee leads to a **new learning paradigm**, called **Dynamic Modeling**.

Motivation

- ▶ In cognitive science, **theory of mind** refers to ones having a model (or models) of another's thoughts, emotions, and perspectives including those different from ones own.
- ▶ Ideally, one might want to have a program (or programs) **generating the behavior** of the other's elicited behavior.
- ▶ But the behavior presented by the other would, in reality, be all and only that **resulting from the interaction between oneself and the other**.
- ▶ While one is attempting to **learn** program(s) for the other, a technique to employ is to pass on a sequence of remarks such as, "I think you are like ...", (where ... might be a program), and, then, attend to the resultantly elicited sequence of reactions of the other as one formulates further programs/models of the other.
- ▶ This active interplay of learner and learnee leads to a **new** learning paradigm, called **Dynamic Modeling**.

Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \quad \square \square \square \dots \square$$

$$q[k]: \quad \square \square \square \dots \square$$

Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \quad \square \square \square \dots \square$$

$$q[k]: \quad \square \square \square \dots \square$$

Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

$$q[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$p[k]:$ \dots

$q[k]:$ \dots

Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

$$q[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

$$q[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

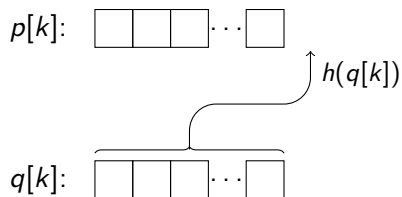
$$p[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

$$q[k]: \quad \overbrace{\boxed{} \boxed{} \boxed{} \cdots \boxed{}}$$

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

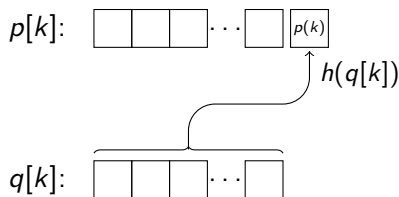
$$\forall k : q(k) := g(p[k])$$



- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$



- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{} \boxed{p(k)}$$

$$q[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{}$$

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \underbrace{\square \square \square \dots \square}_{\text{data}} \square^{p(k)}$$

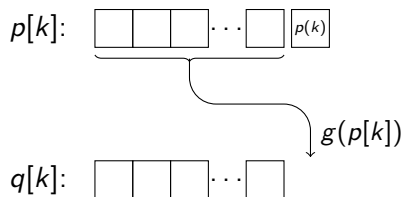
$$q[k]: \square \square \square \dots \square$$

Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

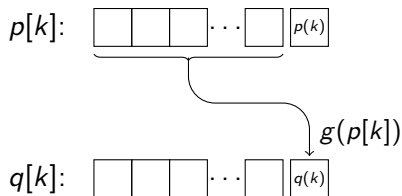


Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$



Crossfeeding

- ▶ Let $\mathbb{N} = \{0, 1, 2, \dots\}$. For total $g : \mathbb{N} \rightarrow \mathbb{N}$ and $\forall k$ let $g[k]$ denote $g(0), \dots, g(k-1)$.
- ▶ **Formally:** Given a computable learner h and a computable learner $g : \mathbb{N} \rightarrow \mathbb{N}$ we define the **learning sequence** p and the **data sequence** q of h on g by

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

$$p[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{} \boxed{p(k)}$$

$$q[k]: \quad \boxed{} \boxed{} \boxed{} \cdots \boxed{} \boxed{q(k)}$$

Crossfeeding

- ▶ In the paper, a wide number of crossfeeding learning criteria are defined. For simplicity, we will present here only a selection, starting with **XBc**.

- ▶ Recall:

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

- ▶ h **XBc-learns** g iff there is i such that $p(i), p(i+1), p(i+2), \dots$ are **programs for** q .
- ▶ h on g tries to find programs for q , the **behavior exhibited by** g , not for g itself.
- ▶ A set \mathcal{S} of total computable functions is **XBc-learnable** iff there exists a computable learner h **XBc-learning** all functions $g \in \mathcal{S}$.

- ▶ In the paper, a wide number of crossfeeding learning criteria are defined. For simplicity, we will present here only a selection, starting with **XBc**.

- ▶ Recall:

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

- ▶ h **XBc**-learns g iff there is i such that $p(i), p(i+1), p(i+2), \dots$ are programs for q .
- ▶ h on g tries to find programs for q , the behavior exhibited by g , not for g itself.
- ▶ A set \mathcal{S} of total computable functions is **XBc-learnable** iff there exists a computable learner h **XBc**-learning all functions $g \in \mathcal{S}$.

- ▶ In the paper, a wide number of crossfeeding learning criteria are defined. For simplicity, we will present here only a selection, starting with **XBc**.

- ▶ Recall:

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

- ▶ h **XBc-learns** g iff there is i such that $p(i), p(i+1), p(i+2), \dots$ are **programs** for q .
- ▶ h on g tries to find programs for q , the **behavior exhibited by** g , not for g itself.
- ▶ A set \mathcal{S} of total computable functions is **XBc-learnable** iff there exists a computable learner h **XBc-learning** all functions $g \in \mathcal{S}$.

- ▶ In the paper, a wide number of crossfeeding learning criteria are defined. For simplicity, we will present here only a selection, starting with **XBc**.

- ▶ Recall:

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

- ▶ h **XBc-learns** g iff there is i such that $p(i), p(i + 1), p(i + 2), \dots$ are **programs for** q .
- ▶ h on g tries to find programs for q , the **behavior exhibited by** g , not for g itself.
- ▶ A set \mathcal{S} of total computable functions is **XBc-learnable** iff there exists a computable learner h **XBc-learning** all functions $g \in \mathcal{S}$.

- ▶ In the paper, a wide number of crossfeeding learning criteria are defined. For simplicity, we will present here only a selection, starting with **XBc**.

- ▶ Recall:

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

- ▶ h **XBc-learns** g iff there is i such that $p(i), p(i+1), p(i+2), \dots$ are **programs for** q .
- ▶ h on g tries to find programs for q , the **behavior exhibited by** g , not for g itself.
- ▶ A set \mathcal{S} of total computable functions is **XBc-learnable** iff there exists a computable learner h **XBc-learning** all functions $g \in \mathcal{S}$.

- ▶ In the paper, a wide number of crossfeeding learning criteria are defined. For simplicity, we will present here only a selection, starting with **XBc**.

- ▶ Recall:

$$\forall k : p(k) := h(q[k])$$

$$\forall k : q(k) := g(p[k])$$

- ▶ h **XBc-learns** g iff there is i such that $p(i), p(i+1), p(i+2), \dots$ are **programs for** q .
- ▶ h on g tries to find programs for q , the **behavior exhibited by** g , not for g itself.
- ▶ A set \mathcal{S} of total computable functions is **XBc-learnable** iff there exists a computable learner h **XBc-learning** all functions $g \in \mathcal{S}$.

Example Results

- ▶ Despite the subtleties of crossfeeding, we have the following two results.
 - ▶ Every single function is (even better than) **XBc**-learnable by infinitely many constant learners.
 - ▶ Every computably enumerable set of functions is (even better than) **XBc**-learnable (this can be shown using infinitary self-reference).
- ▶ In the paper, for a wide class of crossfeeding criteria, we characterize their relation in terms of learning power. **Example immediate** Corollary: **linear time XBc**-learnability is strictly less powerful than **quadratic time XBc**-learnability.

Example Results

- ▶ Despite the subtleties of crossfeeding, we have the following two results.
 - ▶ Every single function is (even better than) **XBc**-learnable by infinitely many constant learners.
 - ▶ Every computably enumerable set of functions is (even better than) **XBc**-learnable (this can be shown using infinitary self-reference).
- ▶ In the paper, for a wide class of crossfeeding criteria, we characterize their relation in terms of learning power. **Example immediate** Corollary: **linear time XBc**-learnability is strictly less powerful than **quadratic time XBc**-learnability.

Example Results

- ▶ Despite the subtleties of crossfeeding, we have the following two results.
 - ▶ Every single function is (even better than) **XBc**-learnable by infinitely many constant learners.
 - ▶ Every computably enumerable set of functions is (even better than) **XBc**-learnable (this can be shown using infinitary self-reference).
- ▶ In the paper, for a wide class of crossfeeding criteria, we characterize their relation in terms of learning power. **Example immediate** Corollary: **linear time XBc**-learnability is strictly less powerful than **quadratic time XBc**-learnability.

Example Results

- ▶ Despite the subtleties of crossfeeding, we have the following two results.
 - ▶ Every single function is (even better than) **XBc**-learnable by infinitely many constant learners.
 - ▶ Every computably enumerable set of functions is (even better than) **XBc**-learnable (this can be shown using infinitary self-reference).
- ▶ In the paper, for a wide class of crossfeeding criteria, we characterize their relation in terms of learning power. **Example immediate** Corollary: **linear time XBc**-learnability is strictly less powerful than **quadratic time XBc**-learnability.

Example Results

- ▶ Despite the subtleties of crossfeeding, we have the following two results.
 - ▶ Every single function is (even better than) **XBc**-learnable by infinitely many constant learners.
 - ▶ Every computably enumerable set of functions is (even better than) **XBc**-learnable (this can be shown using infinitary self-reference).
- ▶ In the paper, for a wide class of crossfeeding criteria, we characterize their relation in terms of learning power. **Example immediate** Corollary: **linear time XBc**-learnability is strictly less powerful than **quadratic time XBc**-learnability.

Cooperative and Secretive Dynamic Modeling

- ▶ **Idea:** As the learner can “see” the learner's outputs, the learner can also try to learn the **elicited behavior** of the learner.
- ▶ **Cooperation:** We say that two functions, h, g , **XBC-cooperate** iff h **XBC-learns** g **and** g **XBC-learns** h .
- ▶ **Secretiveness:** We say that a function h **secretively XBC-learns** a function g iff h **XBC-learns** g **and** g does **not XBC-learn** h .

Cooperative and Secretive Dynamic Modeling

- ▶ **Idea:** As the learner can “see” the learner's outputs, the learner can also try to learn the **elicited behavior** of the learner.
- ▶ **Cooperation:** We say that two functions, h, g , **XBc-cooperate** iff h **XBc-learns** g and g **XBc-learns** h .
- ▶ **Secretiveness:** We say that a function h **secretively XBc-learns** a function g iff h **XBc-learns** g and g does **not XBc-learn** h .

Cooperative and Secretive Dynamic Modeling

- ▶ **Idea:** As the learner can “see” the learner's outputs, the learner can also try to learn the **elicited behavior** of the learner.
- ▶ **Cooperation:** We say that two functions, h, g , **XBc-cooperate** iff h **XBc-learns** g **and** g **XBc-learns** h .
- ▶ **Secretiveness:** We say that a function h **secretively XBc-learns** a function g iff h **XBc-learns** g **and** g does **not XBc-learn** h .

Cooperative and Secretive Dynamic Modeling

- ▶ **Idea:** As the learner can “see” the learner's outputs, the learner can also try to learn the **elicited behavior** of the learner.
- ▶ **Cooperation:** We say that two functions, h, g , **XBc-cooperate** iff h **XBc-learns** g **and** g **XBc-learns** h .
- ▶ **Secretiveness:** We say that a function h **secretively XBc-learns** a function g iff h **XBc-learns** g **and** g does **not XBc-learn** h .

Positive Results Regarding Cooperation

- ▶ There is a computable g so that no computable h that **XBc-learns** g can keep models of its behavior a secret from g , i.e., such h gives itself away: g , in turn, **XBc-learns** h .
- ▶ Positively, such a g is, then, **extremely cooperative**: informally, g can figure out the behavior of any computable h that figures out **its** behavior.
- ▶ Furthermore, such a g can be chosen to be **linear time computable**!

Positive Results Regarding Cooperation

- ▶ There is a computable g so that no computable h that **XBc-learns** g can keep models of its behavior a secret from g , i.e., such h gives itself away: g , in turn, **XBc-learns** h .
- ▶ Positively, such a g is, then, **extremely cooperative**: informally, g can figure out the behavior of any computable h that figures out **its** behavior.
- ▶ Furthermore, such a g can be chosen to be **linear time computable**!

Positive Results Regarding Cooperation

- ▶ There is a computable g so that no computable h that **XBc-learns** g can keep models of its behavior a secret from g , i.e., such h gives itself away: g , in turn, **XBc-learns** h .
- ▶ Positively, such a g is, then, **extremely cooperative**: informally, g can figure out the behavior of any computable h that figures out **its** behavior.
- ▶ Furthermore, such a g can be chosen to be **linear time computable**!

Positive Results Regarding Cooperation

- ▶ There is a computable g so that no computable h that **XBc-learns** g can keep models of its behavior a secret from g , i.e., such h gives itself away: g , in turn, **XBc-learns** h .
- ▶ Positively, such a g is, then, **extremely cooperative**: informally, g can figure out the behavior of any computable h that figures out **its** behavior.
- ▶ Furthermore, such a g can be chosen to be **linear time computable**!

Positive Results Regarding Secretiveness

- ▶ We say that computable h is **extremely uncooperative** iff $\{\text{computable } g \mid h \text{ XBc-learns } g, g \text{ XBc-learns } h\} = \emptyset$.
- ▶ There **are** extremely uncooperative computable h which, nonetheless, are infinitely successful, i.e, such that h **XBc-learns** infinitely many computable g .

Positive Results Regarding Secretiveness

- ▶ We say that computable h is **extremely uncooperative** iff $\{\text{computable } g \mid h \text{ XBC-learns } g, g \text{ XBC-learns } h\} = \emptyset$.
- ▶ There **are** extremely uncooperative computable h which, nonetheless, are infinitely successful, i.e, such that h XBC-learns infinitely many computable g .

Positive Results Regarding Secretiveness

- ▶ We say that computable h is **extremely uncooperative** iff $\{\text{computable } g \mid h \text{ **XBc**-learns } g, g \text{ **XBc**-learns } h\} = \emptyset$.
- ▶ There **are** extremely uncooperative computable h which, nonetheless, are infinitely successful, i.e, such that h **XBc**-learns infinitely many computable g .

Thank You.